AN ABSTRACT OF THE THESIS OF

Paul Oprisan for the degree of Master of Science in

Electrical and Computer Engineering presented on April 6, 1998.

Title: Convergence Study for Adaptive Allpass Filtering

Abstract approved: Redacted for Privacy

Wojtek Kolodziej

Adaptive filtering may be applied in areas where an optimal filtering algorithm may not be known a-priori and where the filtering operation may be non-stationary. This field, or more generally, the field of adaptive systems, is one which may be regarded as mature, having been the subject of considerable research effort in the areas of control and signal processing for almost four decades.

DFE (decision feedback equalization) in various forms has been proposed for detection on magnetic recording channel. An allpass filter is an alternative to the FIR (finite impulse response) forward equalizer which is normally implemented with DFE. This is because the allpass filter is a lower power and complexity alternative, though its behavior and performance are not very well understood yet.

Here, an allpass structure implemented as first and second order IIR (infinite impulse response) filters is examined. Convergence for the LMS (least mean square) adaptation algorithm is studied and, moreover, some convergence conditions and bounds are developed, similarly to the well known FIR case. This thesis provides an useful analytical study of convergence of IIR adaptive filtering. This is accomplished by a systematic approximation of the covariance terms of the adaptive coefficients. The range of the step-size parameter of the LMS algorithm is developed under some simplifying assumptions. All the results obtained are verified by simulation (Matlab and C routines are used).

Convergence Study for Adaptive Allpass Filtering

by

Paul Oprisan

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed April 6, 1998
Commencement June 1998

Master of Science thesis of Paul Oprisan presented on April 6, 1998

APPROVED:

Redacted for Privacy

Major Professor, representing Electrical and Computer Engineering

Redacted for Privacy

Chair of the Department of Electrical and Computer Engineering

Redacted for Privacy

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for Privacy

Paul Oprisan, Author

TABLE OF CONTENTS

LIST OF FIGURES

CONVERGENCE STUDY FOR ADAPTIVE ALLPASS FILTERING

1. INTRODUCTION

Adaptive filtering rapid development over the last thirty years has been made possible by extraordinary advances in the related fields of digital computing, digital signal processing and high speed integrated circuit technology. Practically, adaptive filtering began with research and development efforts in the late 1950's, but the field of adaptive signal processing was established as a distinct discipline in its own right in the 1980's with the publication of the first adaptive signal processing self-contained books by Honig and Messerschmitt in 1984 and Widrow and Stearns in 1985. Nevertheless, one of the earliest publications in adaptive filtering was the paper [2] published in 1960 by Widrow and Hopf that first introduced the least mean squares (LMS) adaptive filtering algorithm. At that time, over ten years before the invention of the microprocessor, digital hardware was not sufficiently advanced for engineers to consider practical implementation of an adaptive filter in purely digital form. Actually, the first experimental filters were implemented as analog circuits with complicated arrangements of analog relays that performed the switching necessary to adjust the filter tap weights.

The simplicity of the least mean square (LMS) algorithm and its robust performance in spite of the simplifying assumptions behind its derivation, attracted the attention of a generation of electrical engineers and formed the basis for intense research and development in adaptive filter architectures and algorithms that continues in force to the present day. The use of adaptive algorithms, in general, and LMS (with a large selection of implementations), in particular, is widespread across varied applications like system identification, adaptive control, transmission systems and adaptive filtering.

In general an adaptive algorithm implies two things: an object upon which processing is carried out (i.e. a control system, transmission system etc.) and the so-called estimation process. The function of an adaptive algorithm is to adjust a parameter vector (generally denoted by θ) with a view to an objective specified by the user; in order to tune this parameter, the user must be able to monitor the system. This task is accomplished via a state vector, generally denoted by X_n where n refers to the time of observation of the system. The rule used to update θ is typically of the form:

$$\theta_n = \theta_{n-1} + \mu_n F(\theta_{n-1}, X_n) \tag{1.1}$$

where μ_n is a sequence of small gains. The choices of the state vector X_n and the function $F(\theta, X)$ are application dependent.

The message that the user desires to send over a telecommunication channel is usually coded to remove redundant information and perhaps allows for the correction of errors at the receiver. The channel will distort the sequence of signals sent and may also add noise. A channel equalizer is a filter whose input is the channel output and whose output is an estimate of the transmitted signal (usually delayed). For a magnetic recording channel the main distortion is produced by intersymbol interference (ISI). ISI cancellation became a crucial problem in the modern magnetic media technology, especially in the view of the increasing demand for higher density disk drives. On the other hand, new technology also requires low access time, which means that the disk is to be read faster. This is an additional source of ISI and of noise, since the spectrum of the transmitted signal extends towards higher frequencies.

Decision feedback equalization has been proposed for detection on magnetic recording channels, mainly because its excellent bit error rate performance at a modest implementation complexity. This is because the forward equalizer, that is normally implemented with DFE, requires multipliers which means it is a major source of complexity and power consumption, in the FIR alternative case. An allpass filter has been shown [1] to be an advantageous structure for the forward equalizer.

The problem of continuous-time adaptive filtering using LMS has been investigated by some authors [6], [7], [12]. There are certain architectural benefits of the continuous-time forward filter over the digital FIR one: sampling can be done at the detector, thereby minimizing the sample delay in both the phase-looked loop (see figure 2.4) and the automatic gain control (not considered in this thesis). Also, a physical implementation uses less die area and, consequently, less power is consumed..

FIR structures have been extensively studied in the literature [3], [4], starting from general FIR structures, or MA models. The goal is to obtain certain convergence conditions and regions of convergence. For example, in the case of a FIR equalizer, the use of LMS as the adaptation algorithm is very common; starting from some assumptions about the statistics of the input and the internal states, a domain of mean square convergence of the adaptive coefficients is derived for the step-size parameter μ_n (in this case $\mu_n = \mu$ is a constant) [3]. Nevertheless, only very few theoretical results were obtained for an allpass structure, or more generally for an IIR structure, mainly because of its strong nonlinear character.

The aim of this thesis is to provide the user of a specific application (DFE for magnetic recording channels) with some useful tools in studying channel equalization. All results are supported by simulation. This can be considered as a starting point to study convergence of adaptive IIR nonlinear filters because the study can be extended from the allpass filter to many IIR nonlinear filters and processes.

First and second order structures for an adaptive allpass filter are presented, with a Lorentzian shape modeling the transition response of the head. The input of the system is a random sequence of dibit responses (presented in section 2.1.), which is simulated by a Matlab procedure. As for any application where decision-directed equalization is employed the error-signal is generated at discrete time instances. Because of the nature of the internal states, this approach implies the sampling of the states of the filter at the same time the input of the decision element in DFE is sampled. Previous work [6] used an exhaustive search to find the optimal pole locations for a low-order allpass filter design

in discrete-time. The present study starts from the canonical form of a forward equalizer which is discretized in order to search for the convergence of the adaptive coefficients.

As mentioned above, there are some simplifying assumptions behind LMS algorithm derivation. These are presented in section 2.2. where a brief overview of FIR adaptive filtering is also presented. Similar assumptions are used in this study for the derivation of the variances of the adaptive coefficients of an allpass filter.

LMS algorithm uses the difference between the actual input of the decision rule and its ideal value [3]. This difference is viewed as an error which is used in adapting the filter feedforward and feedback coefficients.

2. BACKGROUND

2.1. More about magnetic media signal and channel equalization

The reading of a magnetic recording system process is characterized by the step response $s(t)$, shown in figure 2.1. Basically, it represents the response to a positive

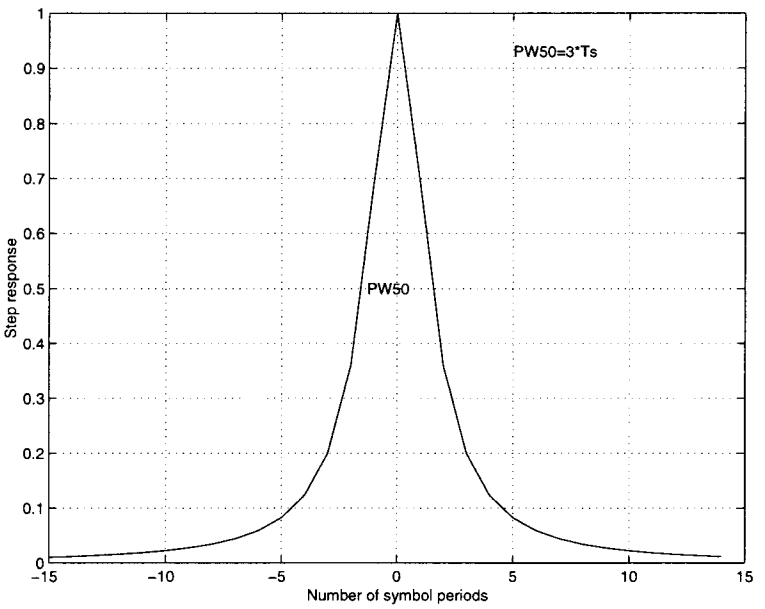

FIGURE 2.1: Read-head step response

transition, from -1 to +1 in the stream of -1 and +1 recorded on the disk. A magnetic recording channel is usually modeled as a linear system and the Lorentzian pulse model for the step response is one of the most common choices:

$$s(t) = \frac{A}{1 + (\frac{2t}{PW50})^2} \tag{2.1}$$

where A is a gain factor, taken 1 in this work, and PW50 is the half-height width of the transition pulse (it is practically a parameter which specifies the density of data on the disk). Because the step response is not convenient to be used in simulation and analysis, the dibit response $p(t)$, shown in figure 2.2, is used instead. The dibit response is the

convolution between s(t) and 1-D, where D is the unit delay:

$$p(t) = s(t) - s(t - T_s), \tag{2.2}$$

where T_s is symbol period. The Fourier transform of $p(t)$ is:

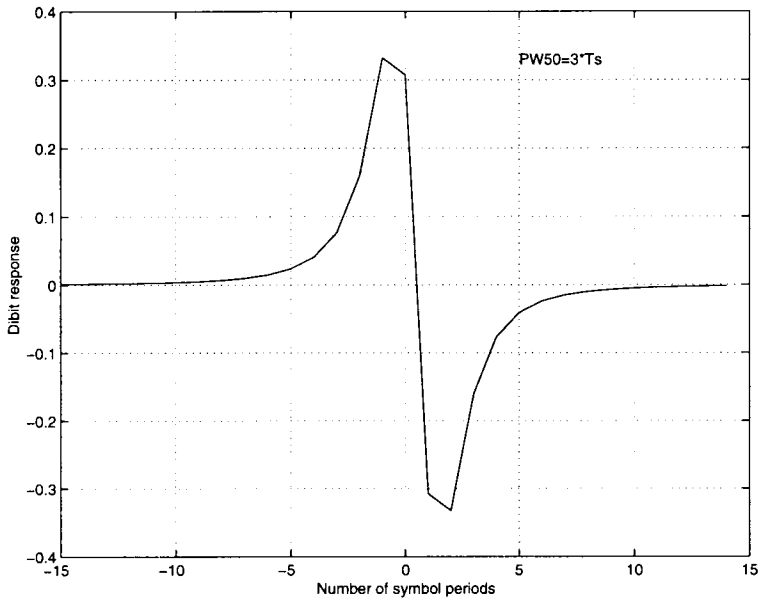

FIGURE 2.2: Read-head dibit response

$$P(w) = S(w)(1 - e^{-jT_s w}), \tag{2.3}$$

where:

$$S(w) = \frac{PW50\pi A}{2} e^{\frac{PW50|w|}{2}} \tag{2.4}$$

is the Fourier transform of the step response, $s(t)$. Thus, the spectrum of the playback signal is bandpass characterized, as shown in figure 2.3.

The spectral energy concentrates at lower frequencies as the symbol density is increased. The receiver must to compromise between the noise power (AWGN in all simulations) and the intersymbol interference (ISI) [8].

The noise in magnetic recording channels consists of media noise, crosstalk between tracks at high data densities and electronic noise and it is common practice to

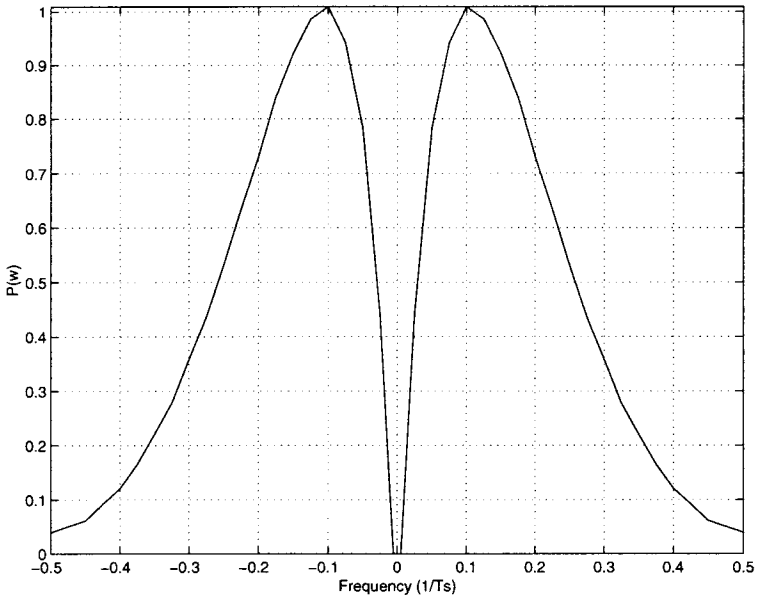

FIGURE 2.3: Dibit frequency response.

consider it additive, white (correlation assumption) and Gaussian (AWGN) for analytical tractability purposes.

A general model of the overall DFE system is given in figure 2.4. The equalizer consists of an allpass filter followed by a lowpass filter. The forward equalizer can be

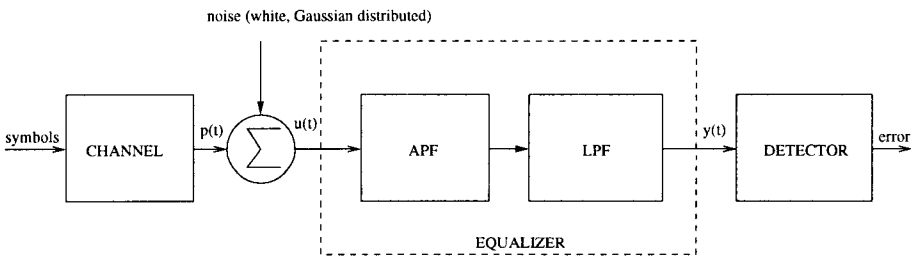

FIGURE 2.4: Block diagram of the DFE system structure used in design

obtained by finding a model of the form:

$$h(t) = q(t) * q(-t), \tag{2.5}$$

where $q(t)$ is the minimum phase and $q(-t)$ is the maximum phase component of $h(t)$, respectively. The corresponding transfer function is:

$$H(s) = Q(s)Q(-s), \tag{2.6}$$

where $Q(-s)$ is modeled only by poles in the right half plane. Any zero in the right half plane would appear as a right half pole in:

$$W(s) = \frac{Q(s)}{Q(-s)}, \tag{2.7}$$

the overall equalizer transfer function. Note that the optimum zero-forcing equalizer given by (2.7) reflects the maximum phase component of the signal to the left half plane. The all pole realization has the form:

$$Q(s) = \frac{K}{1 + \sum_{i=1}^{k} a_i s^i}, \tag{2.8}$$

where k is the order of the filter and $a_i \in (0, \infty)$, $i = 1 \ldots k$.

2.2. FIR adaptive filtering

Considering that $u[n]$ is the input sequence (tap-input vector) of an adaptive filter and $d[n]$ is the desired response of it (the response of the optimal Wiener filter, in many cases), the statistical analysis of the algorithm is carried on starting from the following independence assumptions:

- each sample vector $u[n]$ is statistically independent of all previous sample vectors $u[k]$, $k = 0, 1, \ldots, n-1$,
- each sample vector $u[n]$ is statistically independent of all previous samples of the desired response $d[k]$, which is: $E\{u[n]d^*[k]\} = 0$, $k = 0, 1, \ldots, n-1$,
- the sample $d[n]$ of the desired response is dependent only on the corresponding sample vector $u[n]$ of the input process and statistically independent of all previous samples of the desired response $d[k]$, $k = 0, 1, \ldots, n-1$,

- the tap-input vector and the desired response $d[n]$ consist of mutually Gaussian distributed random vectors for all n.

Assuming that $u[n]$ and $d[n]$ are jointly stationary, the mean square error is given by [3]:

$$J(n) = \sigma_d^2 - a^H(n)\rho - \rho^H a(n) + a^H(n)Ra(n) \tag{2.9}$$

where $a(n)$ is the tap-weight vector, ρ is the cross-correlation vector between $u[n]$ and $d[n]$, R is the correlation matrix of $u[n]$ and $()^H$ denotes the Hermitian transpose (this is in the more general case of complex input signal).

By differentiating the mean squared error $J(n)$ with respect to $u[n]$, we get:

$$\nabla(n) = \frac{\partial J(n)}{\partial a(n)} = -2\rho + 2Ra(n). \tag{2.10}$$

The simplest choice of estimates for R and ρ is to use instantaneous estimates:

$$\hat{R}(n) = u[n]u^H[n] \tag{2.11}$$

$$\hat{\rho}(n) = u[n]d^*[n] \tag{2.12}$$

The instantaneous estimate of the gradient vector is:

$$\hat{\nabla}(n) = -2u[n]d^*[n] + 2u[n]u^H[n]\hat{a}(n). \tag{2.13}$$

At the minimum point of the error-performance surface (defined by $J(n)$ as a function of $a(n)$), the tap-weight vector takes the optimum value a^0, which is given by the normal equation:

$$Ra^0 = \rho \tag{2.14}$$

The tap-weight vector is updated according to:

$$a(n+1) = a(n) + \frac{\mu}{2}(-\nabla(n)) \tag{2.15}$$

where μ is positive constant. Substituting the estimate from (2.13) in this recursive relation, we get:

$$\hat{a}(n+1) = \hat{a}(n) + \mu u[n]\left[d^*[n] - u^*[n]\hat{a}(n)\right]. \tag{2.16}$$

If we denote by $\varepsilon(n) = \hat{a}(n) - a^0$ the weight error vector, by subtracting a^0 from both sides of (2.16), we get:

$$\varepsilon(n+1) = \left[I - \mu u[n]u^H[n]\right]\varepsilon(n) + \mu\left[u[n]d^*[n] - u[n]u^*[n]a^0\right]. \tag{2.17}$$

Because $\hat{a}(n)$ is independent of $u[n]$, it follows that $\varepsilon(n)$ is independent of $u[n]$, so:

$$E\{\varepsilon(n+1)\} = E\{(I - \mu u[n]u^H[n]\varepsilon(n)\} + \mu E\{u[n]d^*[n] - u[n]u^H[n]a^0\}, \tag{2.18}$$

which yields:

$$E\{\varepsilon(n+1)\} = (I - \mu R)E\{\varepsilon(n)\} + \mu(\rho - Ra^0). \tag{2.19}$$

Taking into account the normal equation (2.14), it follows that:

$$E\{\varepsilon(n+1)\} = (I - \mu R)E\{\varepsilon(n)\}. \tag{2.20}$$

This relation gives the necessary condition for *convergence in mean* of the LMS algorithm. Thus, the mean of $\varepsilon(n)$ converges to zero as n approaches infinity for:

$$0 < \mu < \frac{2}{\lambda_{max}}, \tag{2.21}$$

were λ_{max} is the largest eigenvalue of R.

It is useful to develop a recursive relation for the time evolution of the correlation matrix of the weight-error vector:

$$K(n) = E\{\varepsilon(n)\varepsilon^H(n)\}. \tag{2.22}$$

In order to evaluate the correlation matrix $K(n+1)$, we take the expectation of the outer product $\varepsilon(n+1)\varepsilon^H(n+1)$. Algebraic calculations [3] lead to:

$$K(n+1) = K(n) - \mu\left[RK(n) + K(n)R\right] + \mu^2 Rtr[RK(n)] + \mu^2 J_{min}R. \tag{2.23}$$

Note that the last term, $\mu^2 J_{min}R$, prevents $K(n) = 0$ from being a solution to the equation. Thus, $\varepsilon(n)$ only approaches zero, but then executes small fluctuations about zero. It can be shown by induction [3] that $K(n)$ is positive definite. Thus, after each iteration, (2.23)

produces a positive definite answer for the updated value of the weight-error correlation matrix.

The minimum mean square error(MSE) J_{min} is obtained when the coefficient vector $a(n)$ approaches the optimum value a^0, defined by the normal equation. The LMS algorithm relies on a noisy estimate for the gradient vector with the result that the tap-weight vector estimate $\hat{a}(n)$ approaches the optimum value a^0 after a large number of iterations and then executes small fluctuations about a^0. Consequently, the use of the LMS algorithm, after a large number of iterations, results in a mean square error $J(\infty)$ that is greater than the minimum mean square error J_{min}.

As it is shown in [3], the average mean squared error $E\{J(n)\}$ converges to a steady state value equal to $J_{min} + \varepsilon^H(n)R\varepsilon(n)$ if and only if the step-size parameter μ satisfies the condition:

$$0 < \mu < \frac{2}{\sum_{i=1}^{M} \lambda_i}, \tag{2.24}$$

where M is the number of the tap inputs. The above relation gives the condition for the *convergence in mean square* of the LMS algorithm.

A similar condition for the allpass filter cannot be derived because its strong nonlinearity. Actually, updating $K(n)$ using a relation similar to (2.23) is analitycally untractable, so the approach of this study is to derive separate relations for each entry.

Several simulations done before have shown that the LMS algorithm works with good results for an allpass structure. The idea behind this study is to provide the user of DFE applications with some mean square convergence conditions. These conditions are developed in sections 4.1. and 5.2. and all the results are in good agreement with the simulation.

3. ALLPASS FORWARD EQUALIZATION FOR DFE

The optimum equalized response for DFE has a causal time-domain or minimum phase response and it is such that the majority of its energy is represented by the few initial samples. As shown in figure 3.1, the output of the forward equalizer is combined

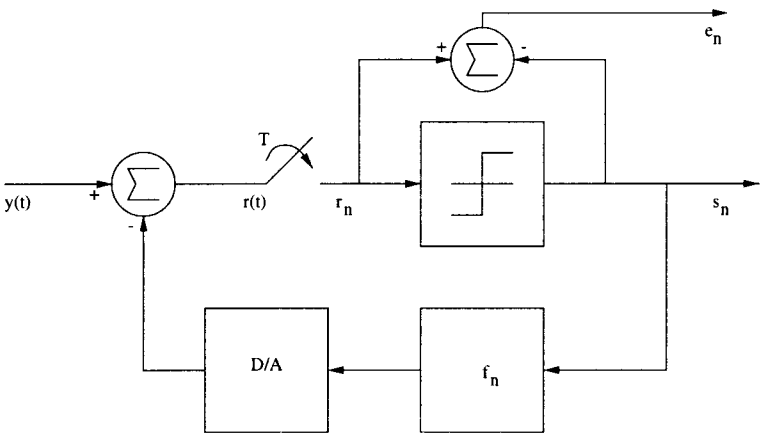

FIGURE 3.1: Detector for adaptive decision feedback equalization.

with the output of the feedback filter in order to obtain the signal:

$$r(t) = y(t) - f_n * s_n, \tag{3.1}$$

where f_n denotes the impulse response of the decision feedback filter and the data $s_n \in \{-1, 1\}$ are the previous decisions.

The sampled signal at the input of the slicer (the decision rule) is:

$$r_n = y_n - f_n * s_n, \tag{3.2}$$

where r_n is determined by sampling $r(t)$ at $nT+\Phi$. T is the sampling period and $\Phi \in (0, T)$ is a phase term which is established by a phase-locked loop using the minimum mean-squared error criterion [9]. The decisions s_n are made by simply taking the sign of r_n. The adaptation algorithm uses the difference between the actual input of the slicer r_n and

its ideal value:

$$e_n = r_n - s_n \tag{3.3}$$

LMS algorithm attempts to minimize the mean square value of the error-signal at the input of the slicer:

$$E\{e_n^2\} = E\Big\{\Big(\sum_{i=0}^{k}(-1)^i b_i x_i - f_n * s_n - s_n\Big)^2\Big\}. \tag{3.4}$$

Usually, the gains of the forward equalizer are updated using the steepest descent algorithm as follows:

$$b_i(n+1) = b_i(n) - \mu\frac{\partial E\{e_n^2\}}{\partial b_i}, \tag{3.5}$$

where μ is a small positive parameter which controls the rate of convergence. Explicitly:

$$b_i(n+1) = b_i(n) - \mu E\{(-1)^i x_i(nT+\phi)e_n\}. \tag{3.6}$$

In practice, the above expectation cannot be obtained and the LMS algorithm, which uses the instantaneous value inside the expectation, is applied:

$$b_i(n+1) = b_i(n) - \mu(-1)^i x_i(nT+\phi)e_n. \tag{3.7}$$

This relation is used in the Matlab simulation of the forward equalizer for updating the coefficients (first and second order filters, see appendices for Matlab functions).

4. FIRST ORDER FORWARD EQUALIZER

4.1. Variance of the adaptive gain

The first order continuous time forward equalizer is given by:

$$W(s) = \frac{-s+a}{s+b}, \tag{4.1}$$

where $a = b$. Applying the bilinear transform (by replacing s with $\frac{1}{2}\frac{1-z^{-1}}{1+z^{-1}}$), the canonical form of the discrete time filter is obtained as:

$$W(z) = \frac{z^{-1}+c}{cz^{-1}+1}, \tag{4.2}$$

where $c = \frac{2a-1}{2a+1}$. The controllable canonical realization of the adaptive filter is shown in figure 4.1.

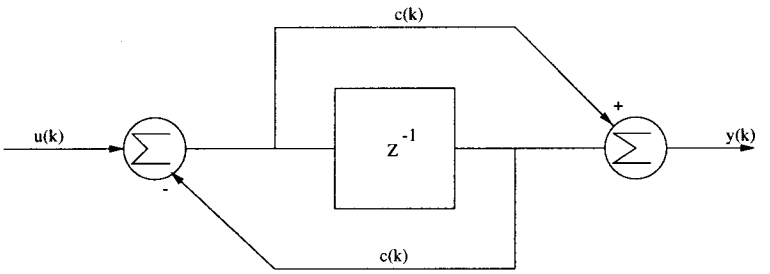

FIGURE 4.1: Discrete time first order filter.

For this simple structure, the state and output equations are:

$$x(k+1) = -c(k)x(k) + u(k) \tag{4.3}$$

and

$$y(k) = [1 - c^2(k)]x(k) + c(k)u(k). \tag{4.4}$$

In order to model the adaptation process, filter's coefficients have to be time-varying, as they adapt towards the optimum MSE solution according to:

$$c(k+1) = c(k) - \mu x(k)\Big((1+c^2(k))x(k) + c(k)u(k) - d(k)\Big). \tag{4.5}$$

By squaring 4.3,

$$x^2(k+1) = c^2(k)x^2(k) - 2c(k)x(k)u(k) + u^2(k), \tag{4.6}$$

and taking the expectation of both sides of (4.6), the second order moment is obtained:

$$E\{x^2(k+1)\} = E\{c^2(k)\}E\{x^2(k)\} - 2E\{c(k)x(k)u(k)\} + E\{u^2(k)\}. \tag{4.7}$$

By squaring (4.5),

$$\begin{aligned} c^2(k+1) = c^2(k) - 2\mu c(k)x(k)\Big((1+c^2(k))x(k) + c(k)u(k) - d(k)\Big) + \\ +\mu^2 x^2(k)\Big((1+c^2(k))x(k) + c(k)u(k) - d(k)\Big)^2, \end{aligned} \tag{4.8}$$

taking expectation of both sides and assuming statistical independence between desired response, state and adaptive coefficient, respectively (similarly to the independence assumptions considered for the FIR case), the variance of the adaptive coefficient is obtained as:

$$\begin{aligned} E\{c^2(k+1)\} = E\{c^2(k)\} - 2\mu E\{c(k) - c^3(k)\}E\{c(k)\}E\{x^2(k)\} - \\ -2\mu E\{c(k)x(k)u(k)\} + \mu^2 E\{[1-c^2(k)]^2\}E\{x^4(k)\} + \mu^2 E\{c^2(k)x^2(k)u^2(k)\} \end{aligned} \tag{4.9}$$

Using the notations K_{xx}, K_{cc} and M_c for $E\{x^2\}$, $E\{c^2\}$ and $E\{c\}$, respectively, we can rewrite (4.7) and (4.9) as:

$$K_{xx}(k+1) = K_{cc}(k)K_{xx}(k) - 2E\{c(k)x(k)u(k)\} + R(k) \tag{4.10}$$

and

$$\begin{aligned} K_{cc}(k+1) = K_{cc}(k) - 2\mu\Big(M_c(k) - E\{c^3(k)\}\Big)K_{xx}(k) - 2\mu E\{c(k)x(k)u(k)\} + \\ +\mu^2 E\{x^4(k)\}\Big(1 - 2K_{cc}(k) + E\{c^4(k)\}\Big) + \mu^2 E\{c^2(k)x^2(k)u^2(k)\}. \end{aligned} \tag{4.11}$$

Considering the input to be zero mean and independent of the internal state and adaptive coefficient (similarly with the FIR case), (4.10) and (4.11) become:

$$K_{xx}(k+1) = K_{cc}(k)K_{xx}(k) + R(k) \tag{4.12}$$

and

$$K_{cc}(k+1) = K_{cc}(k) - 2\mu\Big(M_c(k) - E\{c^3(k)\}\Big)K_{xx}(k)+$$
$$+\mu^2 E\{x^4(k)\}[1 - 2K_{cc}(k) + E\{c^4(k)\}] + \mu^2 K_{cc}(k)K_{xx}(k)R(k). \tag{4.13}$$

In the above relations we have assumed that the state and the adaptive coefficient (gain) are statistically independent, in the sense that a temporal average is considered for the state (fast varying) and a statistical average of the coefficient (slow varying) is used.
The fourth order moment of $x(k)$ in (4.13) can be approximated by (Gaussian approximation, see section 5.2. for a detailed explanation):

$$E\{x^4\} = 2(E\{x^2\})^2. \tag{4.14}$$

Considering $c(k)$ to be zero mean is not a reasonable assumption. However, the third and fourth order moments of $c(k)$ can be approximated by [5](see section 5.2.):

$$E\{c^3(k)\} = M_c(k)\Big(3K_{cc}(k) - 2M_c^2(k)\Big) \tag{4.15}$$

and

$$E\{c^4(k)\} = 3(K_{cc}(k) - M_c^2(k))^2 = 3V_c^2(k), \tag{4.16}$$

where $M_c(k)$ and $V_c(k)$ are the mean and variance of $c(k)$, respectively.
Thus, (4.13) becomes:

$$K_{cc}(k+1) = K_{cc}(k) - 2\mu M_c(k)\Big(1 + 2M_c^2(k) - 3K_{cc}(k)\Big)K_{xx}(k)+$$
$$+2\mu^2 K_{xx}^2(k)\Big(1 - 2K_{cc}(k) + 3V_c^2(k)\Big) + \mu^2 K_{cc}(k)K_{xx}(k)R(k). \tag{4.17}$$

The above recursive relation describes the adaptive coefficient variance behavior for the first order forward equalizer.

4.2. Mean square convergence and experimental results

This section compares the results of a Matlab simulation of a first order forward equalizer with the predicted behavior of the coefficient variance given by (4.17).

The influence of the input noise and the step-size parameter on the convergence of the LMS algorithm is also studied. The former gives information about external parameters influence on the system while the latter allows the user to choose a domain for μ which fits the convergence requirements of a specific application.

The simulation of the first order forward equalizer is performed starting from a given pole location. A sequence of simulated random dibit responses constitutes the input of the channel and AWGN noise is added (with a given variance) in order to simulate the real equalizer input. The forward and feedback coefficients are updated according to (3.7), thus minimizing the output error (see appendices for Matlab functions).

Figures 4.2 and 4.3 present averages of 30 runs of the DFE system. Both the mean and the variance of the coefficient are in good agreement with the simulated behavior of the mean and variance, respectively. These curves are very similar to the usual

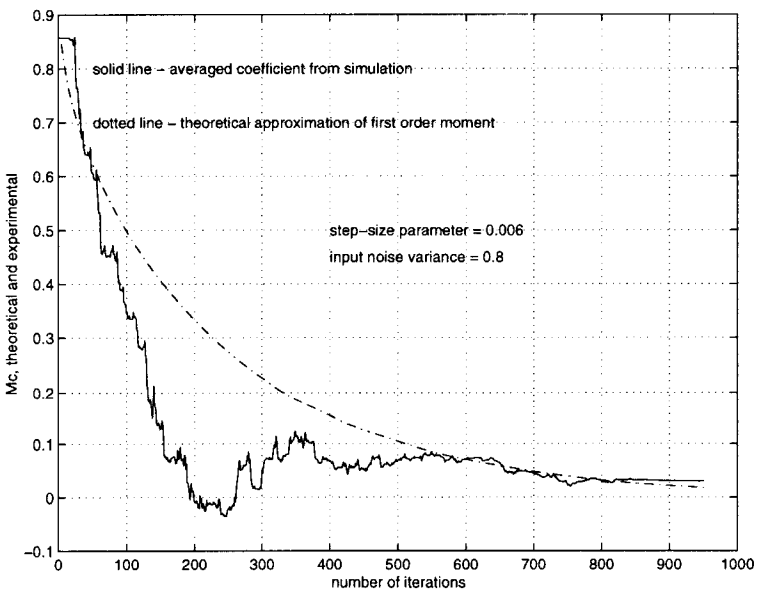

FIGURE 4.2: Mean of the adaptive coefficient

adaptation coefficient error produced by a FIR implementation of the system [3], thus confirming previous results obtained in [6] and [1]. Indeed, both the mean and variance decrease relatively fast (the mean simulation shows a steeper decreasing curve initially,

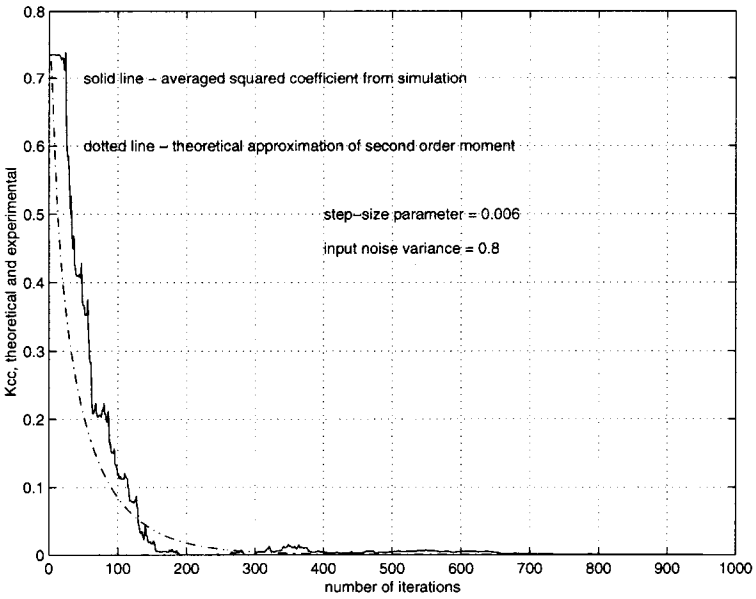

FIGURE 4.3: Variance of the adaptive coefficient

than the predicted behavior of the mean) and then exhibit small fluctuations about the final value.

A question which may arise here is whether the minimum value obtained is a global minimum or just a local one. This issue is addressed by performing several simulations, starting from different initial conditions (i.e. pole locations). In all simulations the final value was the same. Thus, it may be concluded that the solution is 'stable' and therefore acceptable from the practical point of view.

Figure 4.4 shows slight variations of the final value produced by the adaptation for different values of the step-size parameter μ. This is expected because of the similarity with the FIR adaptive filter (using LMS as the adaptive algorithm). In the FIR adaptation case, a necessary and sufficient condition of mean square convergence of the LMS algorithm is given by (2.24). Here, a similar relation cannot be easily derived. Instead, figure 4.5 provides the estimation of the final value of the variance as a function of μ. It can be observed that the variance increases monotonically until convergence is lost. This curve is particular to the given initial conditions, the critical value of μ varies

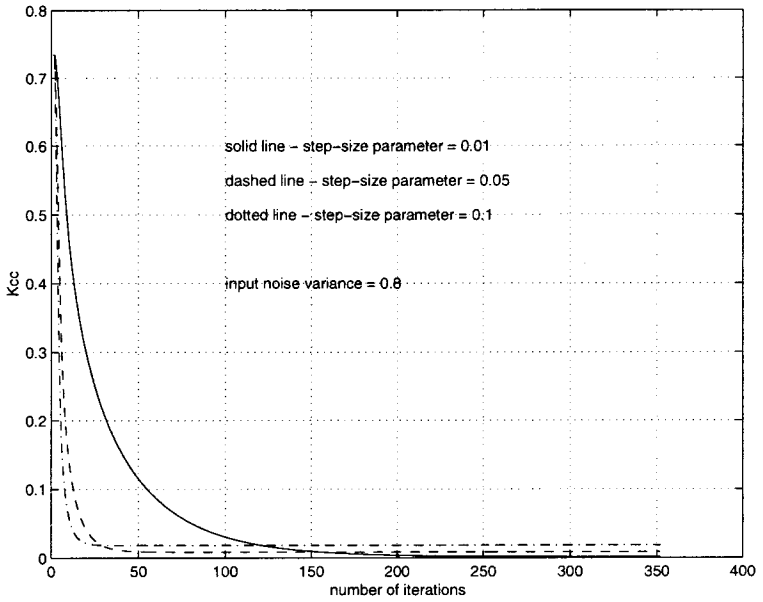

FIGURE 4.4: Variance comparison for different values of μ

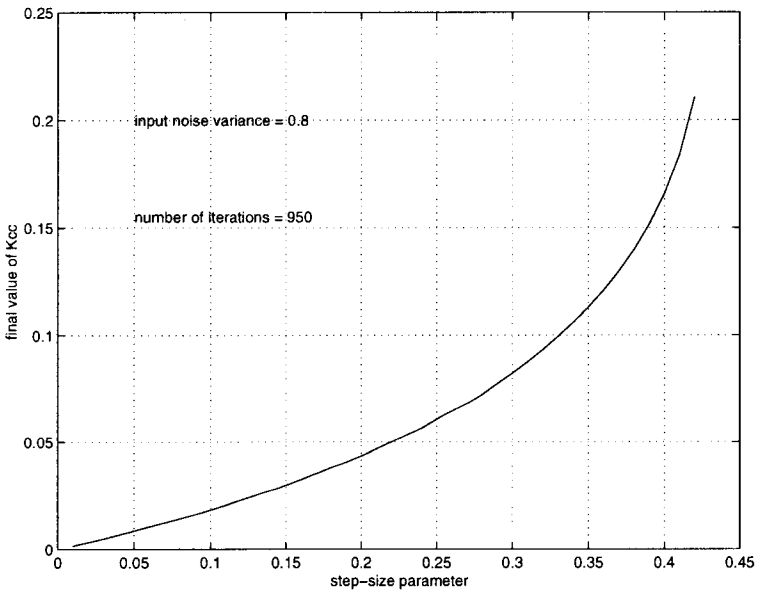

FIGURE 4.5: Variance final value vs. μ

slightly with perturbations in initial conditions. These variations occur for values of μ in the vicinity of the critical one, thus producing variations in the range of μ and do not affect the overall behavior.

The curve from figure 4.5 provides a tool and a theoretical framework at the same time, allowing the allpass DFE user to choose a certain domain of the step-size parameter, in accordance with the application requirements for convergence. It also can be used as a good prediction of the LMS algorithm behavior in adapting the gain.

The input noise power is an important parameter which can affect the behavior of convergence. Indeed, figure 4.6 shows a variation of the final value of the adaptive coefficient with the variance of the input noise for a given value of the step-size parameter. The shape of this curve is somewhat similar to figure 4.5, the final value of the

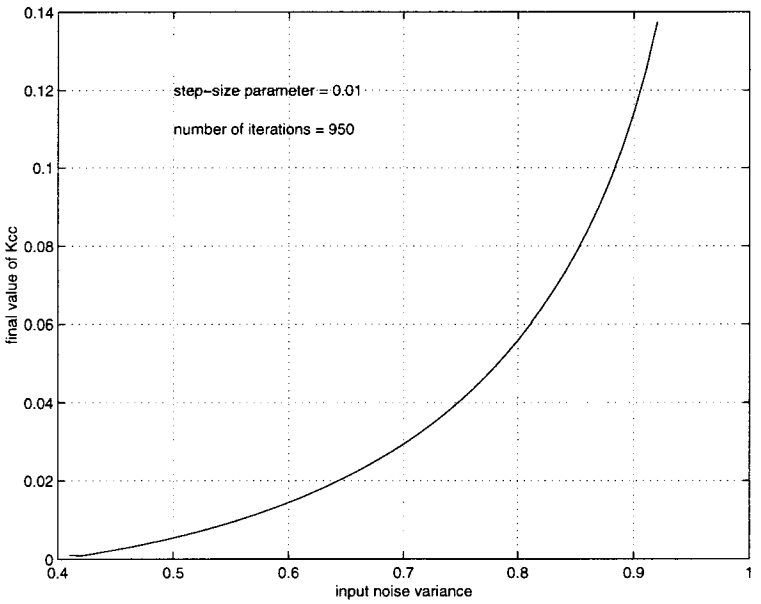

FIGURE 4.6: Variance final value vs. input noise variance

variance increases until a certain value of the input noise variance for which convergence is no longer achieved. This curve provides additional information for designing an allpass forward filter for DFE.

5. SECOND ORDER FORWARD EQUALIZER

5.1. Filter setup

The controllable canonical realization of a second order continuous time filter is given in Figure 5.1. The state and output equations are:

$$\dot{x}(t) = A(t)x(t) + Bu(t) \tag{5.1}$$

$$y(t) = C(t)x(t) - u(t) \tag{5.2}$$

where: A= $\begin{bmatrix} 0 & 1 \\ -a_0 & -a_1 \end{bmatrix}$, B=$\left[0,1\right]^T$ and C=$\left[a_0 - b_0, a_1 + b_1\right]$

The transfer function for the allpass filter is:

$$W(s) = \frac{-s^2 + b_1 s - b_0}{s^2 + a_1 s + a_0}. \tag{5.3}$$

Choosing canonical form for the discretization, which can be obtained by applying the bilinear transform as for the first order filter, we get:

$$W(z) = \frac{-z^{-2} - \frac{8b_0-2}{2b_1+4b_0+1}z^{-1} - \frac{2b_1-4b_0-1}{2b_1+4b_0+1}}{-\frac{2a_1-4a_0-1}{2a_1+4a_0+1}z^{-2} + \frac{8a_0-2}{2a_1+4a_0+1}z^{-1}+1}, \tag{5.4}$$

where $a_i = b_i$, $i = 0, 1$, for the allpass filter. We can rewrite (5.4) as:

$$W(z) = \frac{-z^{-2} - c_1 z^{-1} + c_0}{-c_0 z^{-2} + c_1 z^{-1} + 1}, \tag{5.5}$$

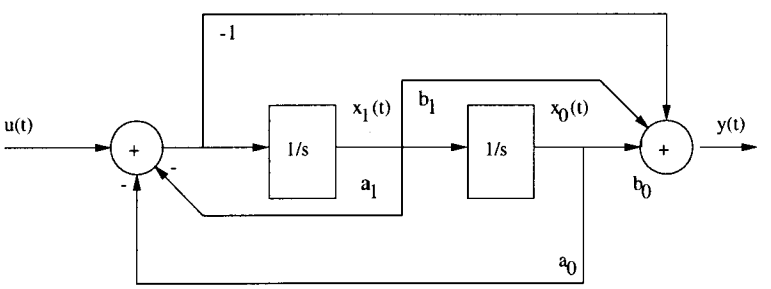

FIGURE 5.1: Continuous time second order filter.

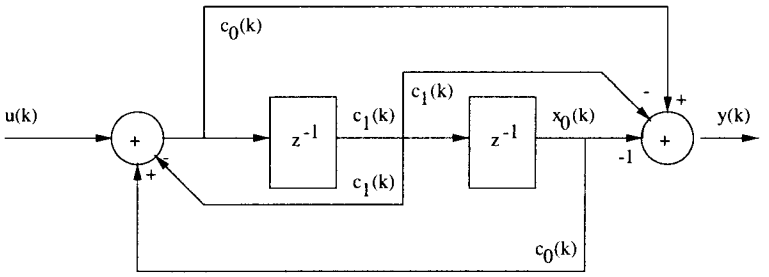

FIGURE 5.2: Discrete time second order filter.

where the coefficients corresponding to the discrete time system are:

$$c_1 = \frac{8b_0 - 2}{2b_1 + 4b_0 + 1} \tag{5.6}$$

$$c_0 = \frac{2b_1 - 4b_0 - 1}{2b_1 + 4b_0 + 1}. \tag{5.7}$$

The discrete time system state and output equations of the adaptive filter are:

$A'(k)= \begin{bmatrix} 0 & 1 \\ c_0(k) & -c_1(k) \end{bmatrix}$, $B' = \left[0,1\right]^T$, $C'(k) = \left[c_0^2(k) - 1, c_1(k)(c_0(k) + 1)\right]$ and $D(k) = c_0(k)$.

The controllable canonical form of the discrete time system is given in Figure 5.2. The state and output equations can be rewritten as:

$$x_0(k+1) = x_1(k) \tag{5.8}$$

$$x_1(k+1) = c_0(k)x_0(k) - c_1(k)x_1(k) + u(k) \tag{5.9}$$

$$y(k) = (c_0^2(k) - 1)x_0(k) - c_1(k)(c_0(k) + 1)x_1(k) + c_0(k)u(k) \tag{5.10}$$

and the filter coefficients are updated according to:

$$c_0(k+1) = c_0(k) -$$

$$-\mu x_0(k)\Big[\Big(c_0^2(k) - 1\Big)x_0(k) + \Big(c_1(k) + c_0(k)c_1(k)\Big)x_1(k) + c_0(k)u(k) - d(k)\Big] \tag{5.11}$$

$$c_1(k+1) = c_1(k) -$$

$$-\mu x_1(k)\Big[\Big(c_0^2(k)-1\Big)x_0(k)+\Big(c_1(k)+c_0(k)c_1(k)\Big)x_1(k)+c_0(k)u(k)-d(k)\Big] \tag{5.12}$$

The recursive relations for updating the coefficients of the filter and the internal states are used in the next section for variances and covariance derivation.

5.2. Covariance matrix

Squaring (5.8) and (5.9), neglecting all terms containing $u(k)$ (we consider the input and the state variables to be statistically independent) and taking expectations, yields:

$$E\{x_0^2(k+1)\}=E\{x_1^2(k)\}, \tag{5.13}$$

$$E\{x_1^2(k+1)\}=E\{c_0^2(k)x_0^2(k)+c_1^2(k)x_1^2(k)-c_0(k)c_1(k)x_0(k)x_1(k)+u^2(k)\} \tag{5.14}$$

and:

$$E\{x_1(k+1)x_0(k+1)\}=E\{c_0(k)x_0(k)x_1(k)-c_1(k)x_1^2(k)\}. \tag{5.15}$$

Similarly, from (5.11) and (5.12) we get:

$$\begin{aligned}E\{c_0^2(k+1)\}=E\{&c_0^2(k)+\mu^2\Big(c_0^2(k)-1\Big)^2x_0^4(k)+\\&+\mu^2\Big(c_1(k)+c_0(k)c_1(k)\Big)^2x_0^2(k)x_1^2(k)+\mu^2c_0^2(k)x_0^2(k)u^2(k)+\\&+2\mu^2\Big(c_0^2(k)-1\Big)\Big(c_1(k)+c_0(k)c_1(k)\Big)x_0^3(k)x_1(k)-2\mu c_0(k)\Big(c_0^2(k)-11\Big)x_0^2(k)-\\&-2\mu c_0(k)\Big(c_1(k)-c_0(k)c_1(k)\Big)x_0(k)x_1(k)\},\end{aligned} \tag{5.16}$$

$$\begin{aligned}E\{c_1^2(k+1)\}=E\{&c_1^2(k)\}+\mu^2\Big(c_0^2(k)-1\Big)^2x_0^2(k)x_1^2(k)+\\&+\mu^2\Big(c_1(k)+c_0(k)c_1(k)\Big)^2x_1^4(k)+\mu^2c_0^2(k)x_1^2(k)u^2(k)+\\&+2\mu^2\Big(c_0^2(k)-1\Big)\Big(c_1(k)+c_0(k)c_1(k)\Big)x_0(k)x_1^3(k)-2\mu c_1(k)\Big(c_0^2(k)-1\Big)x_0(k)x_1(k)-\\&-2\mu c_1(k)\Big(c_1(k)+c_0(k)c_1(k)\Big)x_1^2(k)\}\end{aligned} \tag{5.17}$$

and:

$$E\{c_0(k+1)c_1(k+1)\}=E\{c_0(k)c_1(k)+\mu^2\Big(c_0^2(k)-1\Big)^2x_0^3(k)x_1(k)+$$

$$+\mu^2\Big(c_1(k)+c_0(k)c_1(k)\Big)^2 x_0(k)x_1^3(k)+\mu^2 c_0^2(k)x_0(k)x_1(k)u^2(k)+$$
$$+2\mu^2\Big(c_0^2(k)-1\Big)\Big(c_1(k)+c_0(k)c_1(k)\Big)x_0^2(k)x_1^2(k)-$$
$$-\mu c_1(k)\Big(c_0^2(k)-1\Big)x_0^2(k)-\mu c_1(k)\Big(c_1(k)+c_0(k)c_1(k)\Big)x_0(k)x_1(k)-$$
$$-\mu c_0(k)\Big(c_0^2(k)-1\Big)x_0(k)x_1(k)-\mu c_0(k)\Big(c_1(k)+c_0(k)c_1(k)\Big)x_1^2(k)\} \tag{5.18}$$

With the same assumption as for the first order filter, that states and coefficients are independent, in the sense that the states (fast varying) are temporally averaged and the coefficients (slow varying) are statistically averaged, (5.14) and (5.15) become:

$$E\{x_1^2(k+1)\}=E\{c_0^2(k)\}E\{x_0^2(k)\}+E\{c_1^2(k)\}E\{x_1^2(k)\}-$$
$$-E\{c_0(k)c_1(k)\}E\{x_0(k)x_1(k)\}+E\{u^2(k)\} \tag{5.19}$$

and:

$$E\{x_1(k+1)x_0(k+1)\}=E\{c_0(k)\}E\{x_0(k)x_1(k)\}-E\{c_1(k)\}E\{x_1^2(k)\}. \tag{5.20}$$

Similarly, (5.16), (5.17) and (5.18) become:

$$E\{c_0^2(k+1)\}=E\{c_0^2(k)\}+\mu^2\Big(E\{c_0^4(k)\}-2E\{c_0^2(k)\}+1\Big)E\{x_0^4(k)\}+$$
$$+\mu^2\Big(E\{c_1^2(k)\}+2E\{c_0(k)c_1^2(k)\}+E\{c_0^2(k)c_1^2(k)\}\Big)E\{x_0^2(k)x_1^2(k)\}+$$
$$+\mu^2E\{c_0^2(k)\}E\{x_0^2(k)\}E\{u^2(k)\}-2\mu\Big(E\{c_0^3(k)\}-E\{c_0(k)\}\Big)E\{x_0^2(k)\}+$$
$$+2\mu^2\Big(E\{c_0^2(k)c_1(k)\}-E\{c_1(k)\}+E\{c_0^2(k)c_1(k)\}-E\{c_0(k)c_1(k)\}\Big)E\{x_0^3(k)x_1(k)\}-$$
$$-2\mu\Big(E\{c_0(k)c_1(k)\}+E\{c_0^2(k)c_1(k)\}\Big)E\{x_0(k)x_1(k)\}, \tag{5.21}$$

$$E\{c_1^2(k+1)\}=E\{c_1^2(k)\}+\mu^2\Big(E\{c_0^4(k)\}-2E\{c_0^2(k)\}+1\Big)E\{x_0^2(k)x_1^2(k)\}+$$
$$+\mu^2\Big(E\{c_1^2(k)\}+2E\{c_0(k)c_1^2(k)\}+E\{c_0^2(k)c_1^2(k)\}\Big)E\{x_1^4(k)\}+$$
$$+\mu^2E\{c_0^2(k)\}E\{x_1^2(k)\}E\{u^2(k)\}-2\mu\Big(E\{c_0^2(k)c_1(k)\}-E\{c_1(k)\}\Big)E\{x_0(k)x_1(k)\}+$$
$$+2\mu^2\Big(E\{c_0^2(k)c_1(k)\}-E\{c_1(k)\}+E\{c_0^2(k)c_1(k)\}-E\{c_0(k)c_1(k)\}\Big)E\{x_0(k)x_1^3(k)\}-$$
$$-2\mu\Big(E\{c_1^2(k)\}+E\{c_0(k)c_1^2(k)\}\Big)E\{x_1^2(k)\}, \tag{5.22}$$

and

$$E\{c_0(k+1)c_1(k+1)\} = E\{c_0(k)c_1(k)\}+$$
$$+\mu^2\Big(E\{c_0^4(k)\} - 2E\{c_0^2(k)\} + 1\}\Big)E\{x_0^3(k)x_1(k)\}+$$
$$+\mu^2\Big(E\{c_1^2(k)\} + 2E\{c_0(k)c_1^2(k)\} + E\{c_0^2(k)c_1^2(k)\}\Big)E\{x_0(k)x_1^3(k)\}+$$
$$+\mu^2E\{c_0^2(k)\}E\{x_0(k)x_1(k)\}E\{u^2(k)\} - \mu\Big(E\{c_0^2(k)c_1(k)\} - E\{c_1(k)\}\Big)E\{x_0^2(k)\}-$$
$$-\mu\Big(E\{c_1^2(k)\} + E\{c_0(k)c_1^2(k)\}\Big)E\{x_0(k)x_1(k)\}-$$
$$-\mu\Big(E\{c_0^3(k)\} + E\{c_0(k)\}\Big)E\{x_0(k)x_1(k)\}$$
$$+2\mu^2\Big(E\{c_0^2(k)c_1(k)\} - E\{c_1(k)\} + E\{c_0^3(k)c_1(k)\} - E\{c_0(k)c_1(k)\}\Big)E\{x_0^2(k)x_1^2(k)\}-$$
$$-\mu\Big(E\{c_0(k)c_1(k)\} + E\{c_0^2(k)c_1(k)\}\Big)E\{x_1^2(k)\}. \tag{5.23}$$

We need to approximate the third and fourth order moments in the equations above using lower order moments. The fourth order moments of the states of the system may be evaluated using the Gaussian moment factoring theorem [1] [3]. Thus:

$$E\{x_0^4(k)\} = 2(E\{x_0^2(k)\})^2 \tag{5.24}$$

$$E\{x_0^3(k)x_1(k)\} = 2E\{x_0^2(k)\}E\{x_0(k)x_1(k)\} \tag{5.25}$$

$$E\{x_0^2(k)x_1^2(k)\} = E\{x_0^2(k)\}E\{x_1^2(k)\} + (E\{x_0(k)x_1(k)\})^2 \tag{5.26}$$

In order to evaluate the third and fourth order moments of the adaptive coefficients, we use the following approach: consider a random vector $\eta = (\eta_1, \eta_2)$ and denote by $m_\eta(a,b)$ the moment of order a of η_1 and order b of η_2 and by $s_\eta(a,b)$ the

[1] If $\{u(n)\}$ is a zero-mean complex Gaussian process that is wide-sense stationary and u_n, $n = 1, 2, \ldots, N$ are samples picked from $\{u(n)\}$, then:

$$E[u_{s_1}^* u_{s_2}^* \ldots u_{s_k}^* u_{t_1} u_{t_2} \ldots u_{t_l}] = 0,$$

if $k \neq l$. If N is an even integer and $k = l = N/2$:

$$E[u_{s_1}^* u_{s_2}^* \ldots u_{s_k}^* u_{t_1} u_{t_2} \ldots u_{t_l}] = E[u_{s_{\pi(1)}}^* u_{t_1}]E[u_{s_{\pi(2)}}^* u_{t_2}] \ldots E[u_{s_{\pi(l)}}^* u_{t_l}],$$

where π is a permutation of the set of integers $\{1, 2, \ldots, l\}$ and $\pi(j)$ is the jth element of that permutation.

corresponding cumulant (semi-invariant). The moments we need to approximate are given then by the following relations [2] [5]:

$$m_\eta(2,1) = s_\eta(2,1) + s_\eta(0,1)s_\eta(2,0) + 2s_\eta(1,1)s_\eta(1,0) + s_\eta^2(1,0)s_\eta(0,1), \tag{5.27}$$

$$m_\eta(3,0) = s_\eta(3,0) + s_\eta^3(0,1) + 3s_\eta(1,0)s_\eta(2,0), \tag{5.28}$$

$$m_\eta(4,0) = s_\eta(4,0) + 4s_\eta(3,0)s_\eta(1,0) + 3s_\eta^2(2,0), \tag{5.29}$$

$$\begin{aligned} m_\eta(3,1) = {} & s_\eta(3,1) + s_\eta(3,0)s_\eta(0,1) + 3s_\eta(1,0)s_\eta(1,1) + \\ & + 3s_\eta^2(1,0)s_\eta(1,1) + s_\eta^3(1,0)s_\eta(0,1) + 3s_\eta(2,1)s_\eta(1,0) \end{aligned} \tag{5.30}$$

and

$$\begin{aligned} m_\eta(2,2) = {} & s_\eta(2,2) + 2s_\eta^2(1,1) + s_\eta^2(1,0)s_\eta^2(0,1) + s_\eta^2(1,0)s_\eta(0,2) + \\ & + s_\eta(2,0)s_\eta^2(0,1) + 4s_\eta(1,1)s_\eta(1,0)s_\eta(0,1) + 2s_\eta(2,1)s_\eta(0,1) + \\ & + 2s_\eta(1,2)s_\eta(1,0) \end{aligned} \tag{5.31}$$

In the following we will assume a bivariate Gaussian distribution for the random vector η (which is assumed for the state vector $x = (x_0, x_1)$ and the adaptive coefficient $c = (c_0, c_1)$). The first and second order cumulants are given by [5]:

$$s_\eta(1,0) = m_\eta(1,0) = m_{\eta_1}, \tag{5.32}$$

$$s_\eta(2,0) = m_\eta(2,0) - m_\eta^2(1,0) = var(\eta_1), \tag{5.33}$$

$$s_\eta(1,1) = m_\eta(1,1) - m_\eta(1,0)m_\eta(0,1) = cov(\eta_1, \eta_2) \tag{5.34}$$

---

[2] The general formula connecting moments and cumulants is:

$$m_\eta^{(\nu)} = \sum_{\{r_1\lambda^{(1)} + \ldots + r_x\lambda^{(x)} = \nu\}} \frac{1}{r_1! \ldots r_x!} \frac{\nu!}{(\lambda^{(1)}!)^{r_1} \ldots (\lambda^{(x)}!)^{r_x}} \prod_{j=1}^{x} \left[s_\eta^{(\lambda^{(j)})} \right]^{r_j},$$

where $\sum_{\{r_1\lambda^{(1)} + \ldots + r_x\lambda^{(x)} = \nu\}}$ denotes summation over all unordered sets of different nonnegative integral vectors $\lambda^{(j)}$, $|\lambda^{(j)}| > 0$, and over all ordered sets of positive integral numbers r_j such that $r_1\lambda^{(1)} + \ldots + r_x\lambda^{(x)} = \nu$.

and all the higher order cumulants are zero. With the above notations (5.27),(5.28), (5.29), (5.30) and (5.31) become, respectively:

$$m_\eta(2,1) = var(\eta_1)m_{\eta_2} + 2cov(\eta_1,\eta_2)m_{\eta_1} + m_{\eta_1}^2 m_{\eta_2}, \tag{5.35}$$

$$m_\eta(3,0) = m_{\eta_1}^3 + 3m_{\eta_1}var(\eta_1), \tag{5.36}$$

$$m_\eta(4,0) = 3var^2(\eta_1), \tag{5.37}$$

$$m_\eta(3,1) = 3var(\eta_1)cov(\eta_1,\eta_2) + 3m_{\eta_1}^2 cov(\eta_1,\eta_2) + m_{\eta_1}^3 m_{\eta_2} \tag{5.38}$$

and

$$\begin{aligned} m_\eta(2,2) &= 2cov^2(\eta_1,\eta_2) + m_{\eta_1}^2 m_{\eta_2}^2 + m_{\eta_1}^2 var(\eta_2) + \\ &+ m_{\eta_2}^2 var(\eta_1) + 4cov(\eta_1,\eta_2)m_{\eta_1}m_{\eta_2}. \end{aligned} \tag{5.39}$$

Considering all the above evaluations and using similar notations [3] as for the first order allpass filter, (5.21), (5.22) and (5.23) become:

$$\begin{aligned} K_{c_{00}}(k+1) &= K_{c_{00}}(k) + 2\mu^2 S_0(k)K_{x_{00}}^2(k) + \mu^2 K_{c_{00}}(k)K_{x_{00}}(k)R(k) + \\ &+ \mu^2 S_1(k)\Big(K_{x_{00}}(k)K_{x_{11}}(k) + K_{x_{01}}^2(k)\Big) - 2\mu T_0(k)K_{x_{00}}(k) + \\ &+ 4\mu^2 S_2(k)K_{x_{00}}(k)K_{x_{01}}(k) - 2\mu W_0(k)K_{x_{01}}(k), \end{aligned} \tag{5.40}$$

$$\begin{aligned} K_{c_{11}}(k+1) &= K_{c_{11}}(k) + \mu^2 S_0(k)\Big(K_{x_{00}}(k)K_{x_{11}}(k) + K_{x_{01}}^2(k)\Big) + \\ &+ 2\mu^2 S_1(k)K_{x_{11}}^2(k) + \mu^2 K_{c_{00}}(k)K_{x_{11}}(k)R(k) - 2\mu T_1(k)K_{x_{01}}(k) + \\ &+ 4\mu^2 S_2(k)K_{x_{11}}(k)K_{x_{01}}(k) - 2\mu W_1(k)K_{x_{11}}(k) \end{aligned} \tag{5.41}$$

and

$$\begin{aligned} K_{c_{01}}(k+1) &= K_{c_{01}}(k) + 2\mu^2 S_0(k)K_{x_{00}}(k)K_{x_{01}}(k) + 2\mu^2 S_1(k)K_{x_{11}}(k)K_{x_{01}}(k) - \\ &+ \mu^2 K_{c_{00}}(k)K_{x_{01}}(k)R(k) - \mu T_1(k)K_{x_{00}}(k) - \mu\Big(W_1(k) + T_0(k)\Big)K_{x_{01}}(k) + \\ &+ 2\mu^2 S_3(k)\Big(K_{x_{01}}(k)K_{x_{11}}(k) + K_{x_{01}}^2(k)\Big) - \mu W_0(k)K_{x_{11}}(k), \end{aligned} \tag{5.42}$$

[3] These are: $K_{c_{00}} = E\{c_0^2\}$, $K_{c_{11}} = E\{c_1^2\}$, $K_{c_{01}} = E\{c_0c_1\}$, $K_{x_{00}} = E\{x_0^2\}$, $K_{x_{11}} = E\{x_1^2\}$, $K_{x_{01}} = E\{x_0x_1\}$, $M_{c_0} = E\{c_0\}$ and $M_{c_1} = E\{c_1\}$.

where:

$$S_0(k) = 3K_{c_{00}}^2(k) - 2K_{c_{00}}(k)\Big(3M_{c_0}^2(k)+1\Big) + 3M_{c_0}^4(k) + 1, \tag{5.43}$$

$$S_1(k) = K_{c_{00}}(k)M_{c_1}^2(k) + K_{c_{11}}(k)\Big(M_{c_0}(k)+1\Big)^2 +$$
$$+2K_{c_{01}}(k)\Big(K_{c_{01}}(k)+2M_{c_1}\Big) - M_{c_0}(k)M_{c_1}^2(k)\Big(3M_{c_0}(k)+2\Big), \tag{5.44}$$

$$S_2(k) = 2K_{c_{00}}(k)M_{c_1}(k) + K_{c_{01}}(k)\Big(4M_{c_0}(k)-1\Big) - M_{c_1}(k)\Big(M_{c_0}^2(k)+1\Big), \tag{5.45}$$

$$T_0(k) = M_{c_0}(k)\Big(3K_{c_{00}}(k) - 2M_{c_0}^2(k) - 1\Big) \tag{5.46}$$

$$T_1(k) = K_{c_{00}}(k)M_{c_1}(k) + 2M_{c_0}(k)\Big(K_{c_{01}}(k) - M_{c_0}(k)M_{c_1}(k)\Big) - M_{c_1}(k), \tag{5.47}$$

$$W_0(k) = K_{c_{00}}(k)M_{c_1}(k) + K_{c_{01}}(k)\Big(2M_{c_0}(k)+1\Big) - 2M_{c_0}^2(k)M_{c_1}(k) \tag{5.48}$$

and

$$W_1(k) = K_{c_{11}}(k)\Big(M_{c_0}(k)+1\Big) + 2M_{c_1}(k)\Big(K_{c_{01}}(k) - M_{c_0}(k)M_{c_1}(k)\Big). \tag{5.49}$$

The above relations, together with:

$$M_{c_0}(k+1) = M_{c_0}(k) -$$
$$-\mu\Big(K_{c_{00}}(k)-1\Big)K_{x_{00}}(k) - \mu\Big(K_{c_{01}}(k)+M_{c_1}(k)\Big)K_{x_{01}}(k) \tag{5.50}$$

and:

$$M_{c_1}(k+1) = M_{c_1}(k) -$$
$$-\mu\Big(K_{c_{00}}(k)-1\Big)K_{x_{01}}(k) - \mu\Big(K_{c_{01}}(k)+M_{c_1}(k)\Big)K_{x_{11}}(k) \quad , \tag{5.51}$$

give the complete approximated second order statistics of the adaptive coefficients. The entries in the state covariance matrix are given by the following recursive relations:

$$K_{x_{00}}(k+1) = K_{x_{11}}(k), \tag{5.52}$$

$$K_{x_{11}}(k+1) = K_{c_{00}}(k)K_{x_{00}}(k) + K_{c_{11}}(k)K_{x_{11}}(k) - K_{c_{01}}(k)K_{x_{01}}(k) + R(k), \tag{5.53}$$

and

$$K_{x_{01}}(k+1) = M_{c_0}(k)K_{x_{01}}(k) - M_{c_1}(k)K_{x_{11}}(k). \tag{5.54}$$

Using these approximate recursive relations for the variances, covariance and means of the adaptive coefficients, the MSE convergence is compared with the simulation in the following.

5.3. Mean square convergence of the covariance matrix

This section follows the same line as section 4.2. in comparing the experimental and approximated values of first and second order moments of the adaptive coefficients and analyzing their average behavior as a function of the step-size parameter and the input noise variance.

Figures 5.3 and 5.4 compare the experimental behavior of the LMS algorithm with the approximated curves given by the relations developed in section 5.2.. The initial conditions are the same for both coefficients (in contrast with starting from a given poles location), in order to have a better visualisation of both coefficients mean square convergence. The simulation (there is shown a 30 runs averaged assemble) and the theoretical

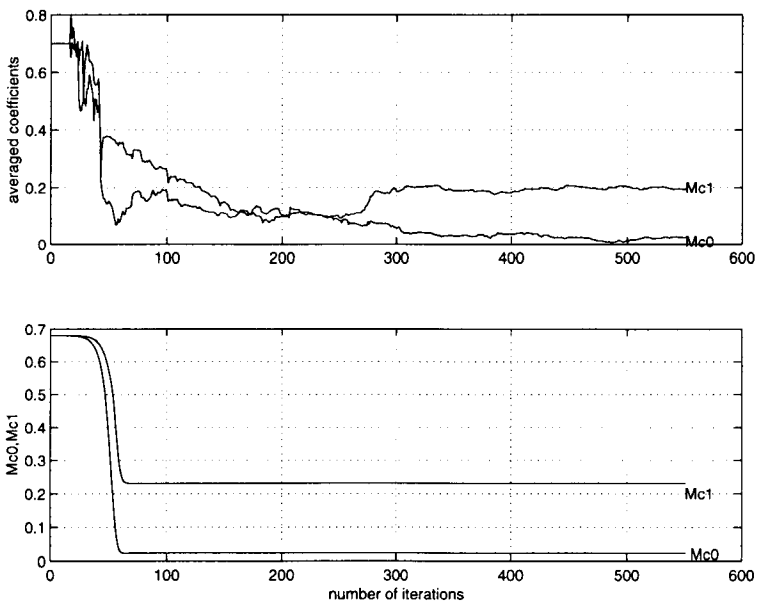

FIGURE 5.3: Means of the adaptive coefficients

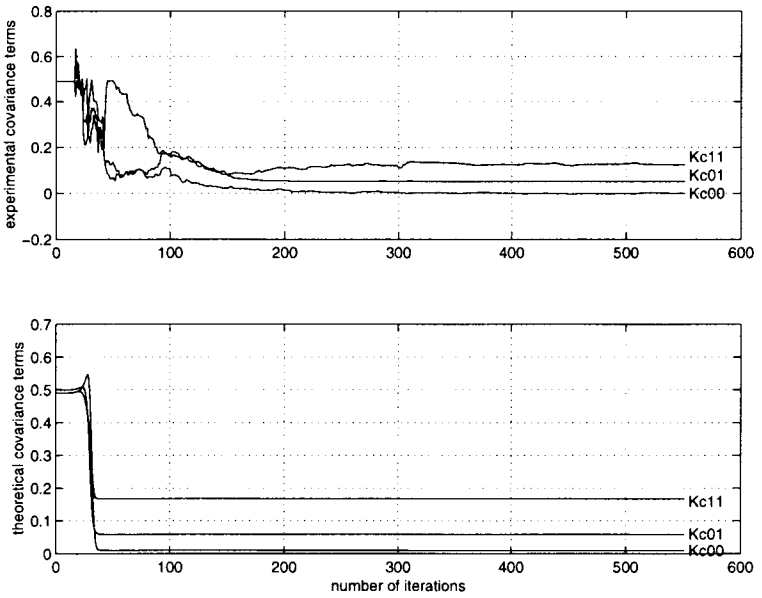

FIGURE 5.4: Variances and covariance of the adaptive coefficients

approximations are performed for the step-size parameter $\mu = 0.005$ and the input noise variance $\sigma = 0.1$

The estimates (upper part of the images) are noisier than in the first order equalizer case, with faster convergence to the final value (which was expected from previous simulation performed in [1] and [6]. The theoretical approximation shows a good agreement with the estimates for both means and variances.

As shown in figure 5.5, the 'turn' corner of all three covariance terms (for relatively large values of the step-size parameter) is sharper than for the first order (figure 4.5), which is closer to a FIR equalizer behavior. As a remark, the 'turn' points, which give the maximum value of μ for which convergence can be achieved, are located at slightly lower values of μ than for the first order. For the FIR equivalent, this is translated by bigger eigenvalues in the second order case than in the first order one. Figure 5.6 is a zoom-in the 'turn' corner region, which is the region of interest for design purposes.

The input noise variance influence on the convergence of LMS algorithm for both the variances and the covariance of the adaptive coefficients is shown in figure 5.7.

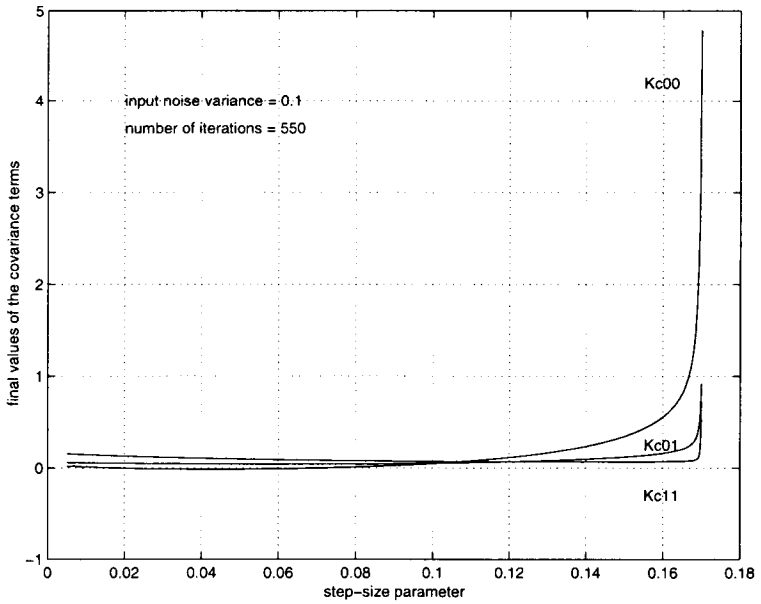

FIGURE 5.5: Final values of variances and covariance vs. μ

The variances of the adaptive coefficients appear more sensitive to noise than the covariance term. On the other hand, the curves show 'turn' points at lower input noise variances than for the first order equalizer, which indicates that the second order filter is more sensitive to noise than the first order one.

In order to better visualize the convergence, especially in the extreme regions, the 3D plots are given for $K_{c_{00}}$ and $K_{c_{11}}$ in figures 5.8 and 5.9. These plots should be helpful in design when there is the need to choose the optimum value of μ (from the speed of convergence point of view) given a certain range for the input noise power.

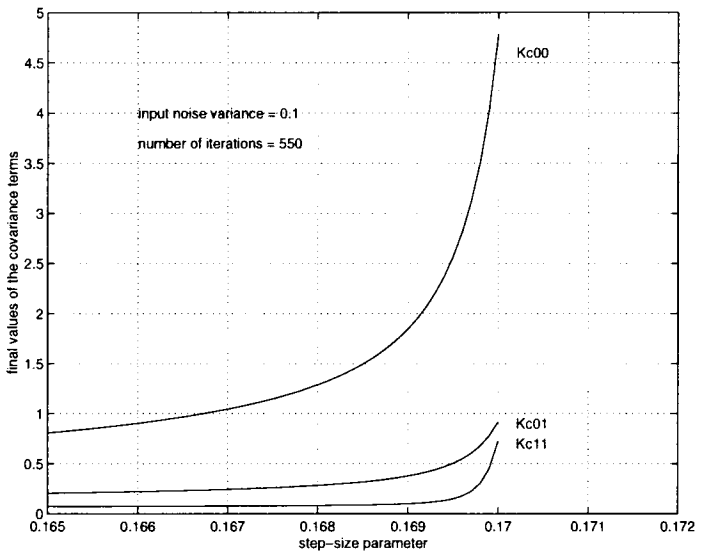

FIGURE 5.6: Final values of $K_{c_{00}}$, $K_{c_{11}}$ and $K_{c_{01}}$ for large μ

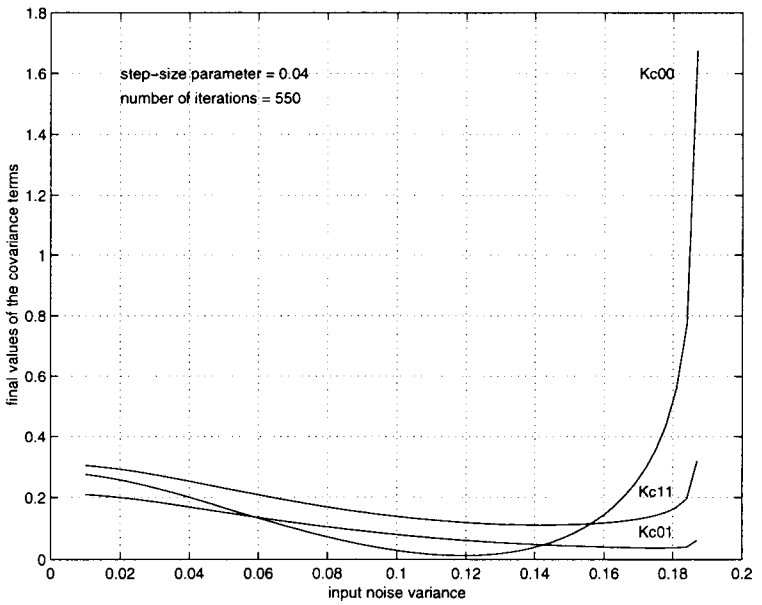

FIGURE 5.7: Final values of variances and covariance vs. σ

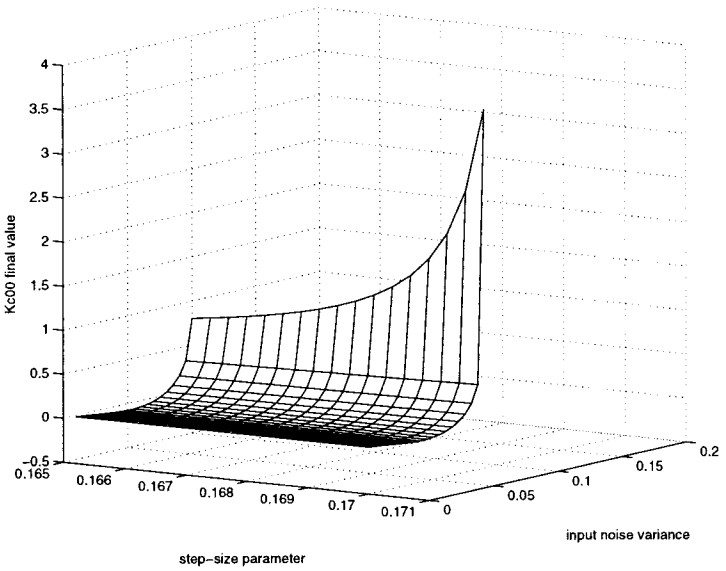

FIGURE 5.8: Surface plot of $K_{c_{00}}$

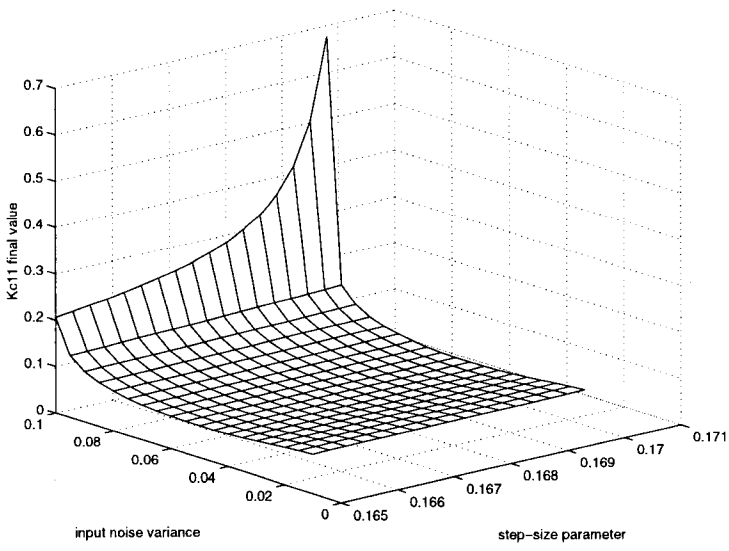

FIGURE 5.9: Surface plot of $K_{c_{11}}$

6. CONCLUSION

Mean square convergence of the adaptive allpass filter, used for a DFE channel forward equalizer is studied. LMS algorithm, or more generally, any stochastic gradient algorithm can be used for adapting the coefficients of an allpass filter. The main reason for using an allpass filter in designing the forward equalizer is based upon practical experience that this type of filter achieves better performance than a regular zero/pole filter. Hence, similar performance can be obtained, at lower implemenation complexity.

The idea behind this study is to provide a theoretical framework for the convergence properties of first and second order allpass filters. The difficulty resides in the filter's strong nonlinear behavior which voids the eigenvalue based method, as in the case of the FIR implementation, and requires nonlinear techniques. A direct derivation of the covariance matrix of the filter coefficients is analytically untractable. Approximate equations for each entry of this matrix are derived in the form of a nonlinear second order system of difference equations. The recursive system of coupled equations is developed under appropriate assumptions concerning the input, the internal states and the adaptive coefficients (some of these approximations are inherited from the well established study of FIR filtering). Finally, these equations are presented in a form suitable for implementation.

Symbolic mathematical software, like Maple or Matematica, could be helpful performing algebraic simplifications, thus reducing the computational burden. Here, its main use is to find the steady state solution in order to verify the simulation results. Maple, used here for the derivation of the covariance terms of the second order filter, also provides the interface with LaTeX, which is a convenient feature for publishing the results. The issue of a powerful design and analysis software package, with links to technical computing environments like Matlab and supporting mathematical software is already in demand in signal processing and control. An example of such package is SaberDesigner.

Enhanced packages should use a large variety of algorithms in adaptive filter design and analysis and the LMS algorithm applied for nonlinear filtering belongs to this set of tools.

In the design of an adaptive system it is important to establish a performance measure that provides comparison of various filter structures and adaptive algorithms in order to select the best solution within the constraints of an application. Some of these measures are: convergence rate, minimum mean square error (MMSE) and misadjustment, the computational complexity, stability and robustness.

A faster convergence does not necessarily implies a better solution. The increased cost and complexity of a faster converging algorithm is only worth the allocation of additional resources if faster convergence is necessary for the system operation. This is the case of DFE for magnetic storage devices such as hard-drives. This problem of fast convergence is addressed in this thesis, in terms of the search for a proper range of the step-size parameter.

The MMSE and the misadjustment (the filter's deviation around the optimal MMSE) are direct measures of how well an adaptive system is able to perform its task. The MMSE depends on many factors, such as: gradient noise, coefficient sensitivities, sensitivities to numerical quantization, the order of the adaptive system, the magnitude of the measurement noise, to name only a few. The coefficient sensitivities and input noise were emphasized in this study.

Low computational complexity is particularly important for real time applications where the algorithm is to be implemented in custom designed VLSI. The allpass filter structure has relatively low complexity. Also, the requirement for an adaptive algorithm to meet performance objectives, while staying computationally simple enough to meet the time constraints of real time operation, justifies the use of LMS algorithm.

In general, the use of IIR filters raises some stability problems, as opposed to the usage of FIR structures which are inherently stable (for proper choices of adaptation gain). Practically, if the poles of an adaptive IIR filter are driven too far outside the unit circle during the adaptation process, the adaptive algorithm itself may become unstable

and the entire adaptation process may diverge. This is a real problem because it was shown experimentally that many IIR filters will achieve faster convergence by allowing their poles to wander outside the unit circle, only to be drawn back towards a stable solution as the adaptive process converges. This study emerges exactly in this shadowed area of adaptive filtering which is not very well understood at the current time.

Robustness is often difficult to measure in a quantitative manner. An important feature of an algorithm is to remain well conditioned regardless of the signal characteristics and to behave well numerically. The first problem was addressed for both first order and second order structure, by studying the influence of the input noise on the final values of the variances of the adaptive coefficients. The simulation shows that the controllable canonical form of the discrete time system does not misbehave numerically.

The theoretical approximations verified by simulation show good convergence properties for the means and variances of the adaptive gains, even for the first order implementation of the allpass filter. These results are in good agreement with previous work reported by Kenney et al. ([1] and [6]). Further development could include similar derivations for a third order implementation of the allpass filter. Additional simplifying assumptions may be needed in order to deal with analytical complexity. One of these assumptions could be to neglect all the terms multiplied by μ^2 in the expressions of the covariance terms. This simplification is based upon the assumption that the third order filter should be even more sensitive to noise than the second order one and the maximum admissible value of the step-size parameter for which convergence is achieved will be smaller.

The adaptive algorithms were implemented as a package of Matlab routines. For the second order filter the routines contain a MEX function written in C language in order to meet the memory and speed requirements. This package is suitable for design purposes and can be easily expanded. Although this convergence study is presented as a part of the decision feedback equalization process (calling for specific parameters and

conditions), it can be easily extended to any application involving nonlinear second order adaptation, such as adaptive echo cancellation, adaptive techniques for audio band noise cancellation and two dimensional filtering of images and video sequences. In the case when adaptive filters need to track rapidly varying signal statistics, LMS algorithm may not be appropriate. However, the moments approximation study can be extended to some adaptive lattice or block adaptive IIR algorithms with reasonable computational complexity.

BIBLIOGRAPHY

1. J. G. Kenney and R. Wood. Multi-level decision feedback equalization, an efficient implementation of FDTS. *IEEE Trans.Magn.*, 31:1115–20, March 1995.

2. B. Widrow and Jr. M. E. Hopf. Adaptive switching circuits. *IRE WESCON Conf.Rec.*, pages 96–104, 1960.

3. S. Haykin. *Adaptive Filter Theory.* Prentice Hall, second edition, 1991.

4. H. J. Kushner and G. G. Yin. *Stochastic Approximation Algorithms and Applications.* Applications of Mathematics. Springer-Verlag, 1997.

5. A. N. Shiryaev. *Probability*, chapter II. Springer-Verlag, second edition, 1996.

6. P. A. McEwen and J. G. Kenney. Allpass forward equalizer for decision feedback recording. *IEEE Trans.Magn*, 31(6):3045–7, November 1995.

7. J. Brown and P. Hurst. Adaptive continuous-time forward equalization for DFE-based disk-drive read channels. 29^{th} *Asimolar Conference on Signals,Systems and Comp.*, 1496:668–72, 1996.

8. G. D. Fourney. Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference. *IEEE Trans.Information Theory*, 18:4194–208, May 1972.

9. J. M. Cioffi, W. L. Abbott, H. K. Thapar, C. M. Melas, and K. D. Fisher. Adaptive equalization in magnetic-disk storage channels. *IEEE Communications Magazine*, pages 15–29, February 1990.

10. W. K. Jenkins et al. *Advanced Concepts in Adaptive Signal Processing.* The Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, 1996.

11. E. A. Lee and D. G. Messerschmitt. *Digital Communication.* Kluwer Academic Publishers, 1988.

12. D. L. Johns, W. M. Snelgrove, and A. S. Sedra. Continuous-time lms adaptive recursive filters. *IEEE Trans. Circuits Syst.*, 38:769–778, July 1991.

APPENDICES

A Matlab functions used in first order filter simulation

```
%Function fAllpass
% equations for mean and variance of the adaptive coefficient
%for first order allpass filter

%initalization of the parameters used in simulation
mu=0.005;
sigmas=0.1;
STEPLEN = 35;
OSR = 1;
PW50 = 3.0;
samplen=950;
pol=0.4;
%input noise variance
Dibit = MakeDibit(OSR,STEPLEN,PW50);
N=length(Dibit);
R=zeros(1,N);
for i=1:N-1
     R(i)=(1/N)*sum(Dibit(1:N-i).*Dibit(i+1:N));
end R0=(1/N)*sum(Dibit(1:N).*Dibit(1:N));
R=[R0 R];
R=R+(sigmas^2)*ones(size(R));
R=[R (sigmas^2)*ones(1,samplen-N)];
% initial conditions
Kxx=0.01;
Mc=Start1a(pol);
Kcc=Mc*Mc;
```

```
% variance and mean iterative relations
for k=1:samplen
     Kxx(k+1)=Kcc(k)*Kxx(k)+R(k);
     Kcc(k+1)=Kcc(k)-2*mu*Mc(k)*(1+2*Mc(k)^2-3*Kcc(k))*Kxx(k)+...
     2*mu^2*(Kxx(k)^2)*(1-2*Kcc(k)+3*(Kcc(k)-...
     Mc(k)^2)^2)+(mu^2^2)*Kcc(k)*Kxx(k)*R(k);
     Mc(k+1)=Mc(k)-mu*(1+Kcc(k))*Kxx(k);
     Kcc=[Kcc Kcc(k)];
     Mc=[Mc Mc(k)];
end

function [NumPass,DenPass,OffPass,CtPass] = Den1Pas()
% this function calculates constants used by function MakeDibit
DenPass = [1 1]';
OffPass = [2 -2]';
NumPass = [2 -2; 1 1];
CtPass = [1 1; -2 2];

function Dibit = MakeDibit(OSR,STEPLEN,PW50)
% Function which provides a dibit response
DiffSig = zeros(OSR+1,1);
DiffSig(1) = 1;
DiffSig(OSR+1) = -1;
MidChan = OSR * ((STEPLEN + 1)/2);
for i=0:MidChan - 1
     Temp = 1 + (4 * i * i)/ ((OSR * PW50) * (OSR * PW50));
     Step1(MidChan + i) = 1/Temp;
```

```
    Step1(MidChan - i) = Step1(MidChan + i);
end
Dibit = conv(DiffSig, Step1);

function Mc=Start1m(pol)
% Function which provides a starting vector a for adaptation
% from discrete-time pole location
xpoly = poly(pol);
bvec = -2-2*xpoly(2);
A=xpoly(2)-1;
Mc = (1/A) * bvec;

function [error,y,alfa,aa,mu1]=FirstOrdp(seedunif,SampLen);
% the function simulates an adaptive allpass filter using
%control canonic structure

%constants used in the simulation
STEPLEN = 35;
DibitLen = 26;
SNR = 20;
Gainnum = 1;
PW50 = 3.0;
OSR = 1;
GAIN = 2.2;
pol = 0.4;
DELAY = 2;
FBLENGTH = 5;
mu = 0.06;
```

```
seedu=924;
SampLen = 950;
Sigma=0.2;
% Order of the equalizer
Order = 1;
% Number of states +1 in the equalizer
LengthXi = Order+1;
% Routine which calculates the functions from which the z-domain
% transfer function can be derived
[ NumPass,DenPass,OffPass,CtPass ] = Den1Pas;
% Function for generating the dibit response of the channel
Dibit = MakeDibit(OSR,STEPLEN,PW50);
% z-doman transfer function
a = Start1a(pol);
aa=a;
% Initialization of the denominator polynomial
den = DenPass * a + OffPass;
% Numerator polynomial in z-domain
num=Gainnum*(-NumPass(1,:)+a*NumPass(2,:));
% adaptation process
y(DELAY+FBLENGTH) = 0;
b = y(DELAY:DELAY+FBLENGTH);
b1 = [zeros(1,DELAY-1),b];
for i=1:FBLENGTH+1
     brev(i) = b(FBLENGTH + 2 - i);
end
mu1 = mu;
rand('seed', seedunif);
```

```
datin = rand(SampLen,1) - 0.5;
bindat = sign(datin);
beqout1 = conv(bindat,b1);
chanout = conv(Dibit, bindat);
noise = Sigma*Sigma*randn(SampLen,1);
% initializing ouput error
error = zeros(SampLen,1);
yout = zeros(SampLen,1);
ydiff = zeros(SampLen,1);
ystate = zeros(SampLen,1);
xi = zeros(LengthXi,1);
adjst0 = zeros(LengthXi,1);
ydi = zeros(LengthXi,1);
fbdata = zeros(FBLENGTH+1,1);
alfa = a;
for i=1:SampLen
% Evaluation of the recursion portion of the discrete-time filter
     xi(1) = (-den(2:LengthXi)'*xi(2:LengthXi)+GAIN*...
     (noise(i)+chanout(i)))/den(1);
     ctstate = CtPass * xi;

     %feedback adaptation
     if i > (DELAY+FBLENGTH)
          fbdata = bindat(i-DELAY-FBLENGTH+1:i-DELAY+1);
     end
     beqout(i) = brev * fbdata;
     %Summing the scaled states of the forward equalizer to
```

```
      %the output of the feedback filter
      yout(i) = ctstate(LengthXi) + a(1:Order)'*...
      ctstate(1:Order) - beqout(i);
      if i > (DELAY+FBLENGTH)
           error(i) = yout(i) - bindat(i-DELAY+2);
      end
      % Update the feedback filter
      brev = brev + (mu * error(i) * fbdata');
      a = a - mu1 * error(i) * (ctstate(1:Order));
      alfa =[alfa a];
      xi(2:LengthXi) = xi(1:Order);
      adjst0(1) = (-den(2:LengthXi)'*adjst0(2:LengthXi)+...
      2*ctstate(1))/den(1);
      ctadjst0 = CtPass * adjst0;
      sgrad(i) = ctadjst0(2) + a * ctadjst0(1);
      stateval(i) = ctstate(1);
      adjst0(2:LengthXi) = adjst0(1:Order);
      den = DenPass * a + OffPass;
end
seedunif = rand('seed');

% Function StFirst which calculates the adaptive vectors
seedu=924;
SampLen = 950;
no=30;
% initalization of the adaptation vector
beta1=zeros(1,SampLen+1);
```

```
% initialization of the output error
errorstat=zeros(Sampen,1);
% the adaptation vector is updated
for k=1:no
     [error,y,alfa,aa,mu1] = FirstOrdp(seedu,SampLen);
     errorstat = errorstat + error.*error;
     beta1=[beta1; alfa];
end
gamma1=beta1(2:k+1,:);
coef1=mean(gamma1);
```

B Matlab functions used in second order filter simulation

```
% Function sAllpass
% equations for means and variances of the adaptive coefficients
%for second order allpass filter

% initialization of the parameters used in simulation
sigmas=0.1;
STEPLEN = 35;
OSR = 1;
PW50 = 3.0;
samplen=550;
DELAY=17;
FBLENGTH=15;
pol=[0.4 0.4];
% input noise variance
Dibit = MakeDibit(OSR,STEPLEN,PW50);
N=length(Dibit);
R=zeros(1,N);
for i=1:N-1 R(i)=sum(Dibit(1:N-i).*Dibit(i+1:N));
end R0=sum(Dibit(1:N).*Dibit(1:N));
R=[R0 R];
R=R+(sigmas^2)*ones(size(R));
R=[R (sigmas^2)*ones(1,samplen-N)];
mu=0.00001;
% inital conditions
mc0=0.7;
```

```
mc1=0.7;
kc00=0.49;
kc11=0.49;
kc01=0.49;
% iterative equations
Kx00=[0.1 zeros(1,samplen)];
Kx01=[0.1 zeros(1, samplen)];
Kx11=[0.1 zeros(1, samplen)];
Kc00=[kc00 zeros(1, samplen)];
Kc01=[kc01 zeros(1, samplen)];
Kc11=[kc11 zeros(1, samplen)];
M0=[mc0 zeros(1, samplen)];
M1=[mc1 zeros(1, samplen)];
K=[Kx00; Kx11; Kx01; M0; M1; Kc00; Kc11; Kc01; R]';
% call the computing MEX function
kk=CovMatrix(K,mu);

% Function sAllpassMu
% using the same equations as function sAllpass, gives the final values
% of the second order moments vs.  step-size parameter
sigmas=0.1;
STEPLEN = 35;
OSR = 1;
PW50 = 3.0;
samplen=550;
DELAY=17;
FBLENGTH=15;
Dibit = MakeDibit(OSR,STEPLEN,PW50);
```

```
N=length(Dibit);
R=zeros(1,N);
for i=1:N-1 R(i)=sum(Dibit(1:N-i).*Dibit(i+1:N));
end R0=sum(Dibit(1:N).*Dibit(1:N));
R=[R0 R];
R=R+(sigmas^2)*ones(size(R));
R=[R (sigmas^2)*ones(1,samplen-N)];
kvsmu=[0 0 0];
mc0=0.7;
mc1=0.7;
kc00=0.49;
kc11=0.49;
kc01=0.49;

     Kx00=[0.1 zeros(1,samplen)];
Kx01=[0.1 zeros(1, samplen)];
Kx11=[0.1 zeros(1, samplen)];
Kc00=[kc00 zeros(1, samplen)];
Kc01=[kc11 zeros(1, samplen)];
Kc11=[kc01 zeros(1, samplen)];
M0=[mc0 zeros(1, samplen)];
M1=[mc1 zeros(1, samplen)];
K=[Kx00; Kx11; Kx01; M0; M1; Kc00; Kc11; Kc01; R]';
for mu=0.165:0.0001:0.17      kk=CovMatrix(K,mu);
     finals=[kk(samplen,6:8)];
     kvsmu=[kvsmu; finals];
end
kvsmu=kvsmu(2:size(kvsmu,1),:);
```

```
function [NumPass,DenPass,OffPass,CtPass] = Den2Pas()
% this function calculates constants used by function MakeDibit
DenPass = [1 2 ; 2 0; 1 -2];
OffPass = [4; -8; 4];
NumPass = [4 -8 4; 2 0 -2; 1 2 1];
CtPass = [1 2 1; -2 0 2; 4 -8 4];

%Function CovMatrix (C language) which computes the covariance matrix entries

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "mex.h"
#define elem(i,j) pr[i+j*k]

/* Computational routine for the covariance matrix*/

     void covmat(double* pr, int k,double mu)
{
     int i;
     double s0,s1,s2,t0,t1,w0,w1;

for (i=0; i<k-1; i++)
     {
     s0=3*pow(elem(i,5),2)-2*elem(i,5)*(3*pow(elem(i,3),2)+1)+
     3*pow(elem(i,3),4)+1;
```

```
s1=elem(i,5)*pow(elem(i,4),2)+elem(i,6)*pow(elem(i,3)+1,2)+
2*elem(i,7)*(elem(i,7)+
2*elem(i,4))-elem(i,3)*pow(elem(i,4),2)*(3*elem(i,3)+2);
s2=2*elem(i,5)*elem(i,4)+elem(i,7)*(4*elem(i,4)-1)-
elem(i,4)*(pow(elem(i,3),2)+1);
t0=elem(i,3)*(3*elem(i,5)-2*pow(elem(i,3),2)+1);
t1=elem(i,5)*elem(i,4)+2*elem(i,3)*(elem(i,7)-
elem(i,3)*elem(i,4))+elem(i,4);
w0=elem(i,5)*elem(i,4)+elem(i,7)*(2*elem(i,3)+1)-
2*pow(elem(i,3),2)*elem(i,4);
w1=elem(i,6)*(elem(i,3)+1)+2*elem(i,4)*(elem(i,7)-
elem(i,3)*elem(i,4));
elem(i+1,0)=elem(i,1);
elem(i+1,1)=elem(i,5)*elem(i,0)+elem(i,6)*elem(i,1)-
elem(i,7)*elem(i,2)+elem(i,8);
elem(i+1,2)=elem(i,3)*elem(i,2)-elem(i,4)*elem(i,1);
elem(i+1,3)=elem(i,3)+mu*(elem(i,5)-1)*elem(i,0)-
mu*(elem(i,4)+elem(i,7))*elem(i,2);
elem(i+1,4)=elem(i,4)-mu*(elem(i,5)-1)*elem(i,2)-
mu*(elem(i,4)+elem(i,7))*elem(i,1);
elem(i+1,5)=elem(i,5)+2*pow(mu,2)*s0*pow(elem(i,0),2)+
pow(mu,2)*elem(i,5)*elem(i,0)*elem(i,8)+pow(mu,2)*s1*
(elem(i,0)*elem(i,1)+pow(elem(i,2),2))-2*mu*t0*elem(i,0)+
4*pow(mu,2)*s2*elem(i,0)*elem(i,2)-2*mu*w0*elem(i,2);
elem(i+1,6)=elem(i,6)+pow(mu,2)*elem(i,5)*elem(i,1)*elem(i,8)+
pow(mu,2)*s0*(elem(i,0)*elem(i,1)+pow(elem(i,2),2))+
2*pow(mu,2)*s1*pow(elem(i,1),2)-2*mu*t1*elem(i,2)+
```

```
      4*pow(mu,2)*s2*elem(i,1)*elem(i,2)-2*mu*w1*elem(i,1);
      elem(i+1,7)=elem(i,7)+pow(mu,2)*elem(i,5)*elem(i,2)*elem(i,8)+
      2*pow(mu,2)*s0*elem(i,0)*elem(i,2)+2*pow(mu,2)*s1*elem(i,1)*
      elem(i,2)-mu*t1*elem(i,0)-mu*(w1+t0)*elem(i,2)+
      2*pow(mu,2)*s2*(elem(i,0)*elem(i,1)+pow(elem(i,2),2))-
      mu*w0*elem(i,1);
      }
}
/* Gateway routine */
void mexFunction(int nlhs, Matrix*plhs[], int nrhs, Matrix*prhs[])

      unsigned int m,n;
      double *muget;
      double stepsize;
      Matrix *ktp;
/* Size of the output covariance matrix*/
m=mxGetM(prhs[0]);
n=mxGetN(prhs[0]);
/* Create matrix for return argument*/
ktp=mxCreateFull(m,n,REAL);
memcpy((char*)mxGetPr(ktp),(char*)mxGetPr(prhs[0]),sizeof(double)*m*n);
/*Dereference arguments*/
muget=mxGetPr(prhs[1]);
stepsize=muget[0];
/*Call the operating function*/
covmat(mxGetPr(ktp),m,stepsize);
/*Return the new matrix*/
```

```
plhs[0]=ktp;
}
function [error,alfa,aa,mu1] = SecondOrdp(seedunif,SampLen);
%adaptive allpass filter using control canonic structure

% constants used in the simulation
STEPLEN = 35;
DibitLen = 26;
SNR = 20;
Gainnum = 1;
PW50 = 3.0;
OSR = 1;
Sigma =0.08;
GAIN = 2.2;
pol = [0.2 0.2];
DELAY = 5;
FBLENGTH = 10;
mu = 0.005;
% Order of the equalizer
Order = 2;
% Number of states +1 in the equalizer
LengthXi = Order+1;
% Routine which calculates the functions from which the z-domain
% transfer function can be derived
[ NumPass,DenPass,OffPass,CtPass ] = Den2Pas;
% Function for generating the dibit response of the channel
Dibit = MakeDibit(OSR,STEPLEN,PW50);
a=[0.7 0.7]';
```

```
aa=a;
% Initialization of the denominator polynomial
den = DenPass * a + OffPass;
% Numerator polynomial in z-domain
num=Gainnum*(-NumPass(1,:)+a(2)*NumPass(2,:)-a(1)*NumPass(3,:));
% adaptation process
y(DELAY+FBLENGTH) = 0;
b = y(DELAY:DELAY+FBLENGTH);
b1 = [zeros(1,DELAY-1),b];
for i=1:FBLENGTH+1
     brev(i) = b(FBLENGTH + 2 - i);
end
mu1 = mu;
rand('seed',seedunif);
datin = rand(SampLen,1) - 0.5;
bindat = sign(datin);
beqout1 = conv(bindat,b1);
chanout = conv(Dibit, bindat);
noise = (Sigma^2)*randn(SampLen,1);
% initializong output error
error = zeros(SampLen,1);
yout = zeros(SampLen,1);
ydiff = zeros(SampLen,1);
ystate = zeros(SampLen,1);
xi = zeros(LengthXi,1);
adjst0 = zeros(LengthXi,1);
ydi = zeros(LengthXi,1);
fbdata = zeros(FBLENGTH+1,1);
```

```
alfa = a;
for i=1:SampLen
% Evaluate the recursion portion of the discrete-time filter
     xi(1) = (-den(2:LengthXi)'*xi(2:LengthXi)+...
     GAIN*(noise(i)+chanout(i)))/den(1);
     ctstate = CtPass * xi;
     tapnorm=(abs(ctstate(1)))^2+(abs(ctstate(2)))^2;
     munorm=mu1/tapnorm;
     % feedback adaptation
     if i > (DELAY+FBLENGTH)
          fbdata = bindat(i-DELAY-FBLENGTH+1:i-DELAY+1);
     end
     beqout(i) = brev * fbdata;
     % Summing the scaled states of the forward equalizer to the output
     yout(i) = ctstate(LengthXi) + a(1:Order)'*...
     ctstate(1:Order) - beqout(i);
     if i > (DELAY+FBLENGTH)
          error(i) = yout(i) - bindat(i-DELAY+2);
     end
     % Update the feedback filter
     brev = brev + (mu * error(i) * fbdata');
     a = a - munorm * error(i) * (ctstate(1:Order));
     alfa =[alfa a];
     xi(2:LengthXi) = xi(1:Order);
     adjst0(1) = (-den(2:LengthXi)'*adjst0(2:LengthXi)+...
     2*ctstate(2))/den(1);
     ctadjst0 = CtPass * adjst0;
     sgrad(i) = ctadjst0(3) + a(2) * ctadjst0(2);
```

```
      stateval(i) = ctstate(2);
      adjst0(2:LengthXi) = adjst0(1:Order);
      den = DenPass * a + OffPass;
end
% final impulse response
num=Gainnum*(-NumPass(1,:)+a(2)*NumPass(2,:)-a(1)*NumPass(3,:));
y = GAIN * filter(num,den,Dibit);
seedunif = rand('seed');

%Function StSecond which calculates the adaptive vectors
seedu =524;
SampLen = 550;
% initialization of the output error
errorstat = zeros(SampLen,1);
no=30;
%initialization of the adaptation vectors
beta1=zeros(1,SampLen+1);
beta2=zeros(1,SampLen+1);
% the adaptation vectors are updated
for k=1:no
      [error,alfa,aa,mu1] = SecondOrdp(seedu,SampLen);
      errorstat = errorstat + error.*error;
      beta1=[beta1; alfa(1,:)];
      beta2=[beta2; alfa(2,:)];
end
gama1=beta1(2:k+1,:);
gama2=beta2(2:k+1,:);
```

```
coef1=mean(gama1);

coef2=mean(gama2);

kcoef00=mean(gama1.*gama1);

kcoef11=mean(gama2.*gama2);

kcoef01=mean(gama1.*gama2);
```

INDEX